Teaching Algebra through Functional Programming:An Analysis of the Bootstrap Curriculum

Robert Lee
Brigham Young University - Provo

Teaching Algebra Through Functional Programming:

An Analysis of the Bootstrap Curriculum

Robert William Lee

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

School of Technology

Brigham Young University

March 2013

ABSTRACT

Teaching Algebra Through Functional Programming: An Analysis of the Bootstrap Curriculum

Robert Lee
School of Technology, BYU
Master of Science

Bootstrap is a computer-programming curriculum that teaches students to program video games using Racket, a functional programming language based on algebraic syntax. This study investigated the relationship between learning to program video games from a Bootstrap course and the resulting effect on students' understanding of algebra. Courses in three different schools, lasting about six weeks each, were studied. Control and treatment groups were given a pre and post algebra assessment. A qualitative component consisting of observations and interviews was also used to further triangulate findings. Statistical analysis revealed that students who completed the Bootstrap course gained a significantly better understanding of variables and a suggestive improvement in understanding functions. In the assessments, students failed to demonstrate a transfer of the advanced concepts of function composition and piecewise functions from programming to algebraic notation. Interviews with students demonstrated that with coaching, students were able to relate functions written in Racket to functions written in algebraic notation, but were not yet able to transfer their experience of function composition from programming to algebra.

Keywords: Bootstrap, Racket, functional programming, algebra, functions, variables

ACKNOWLEDGMENTS

First my appreciation to the Bootstrap research team at BYU, who put together the Bootstrap courses and associated research. Thanks to the professors, who also made up my committee; the instructors of the Bootstrap courses, Anna McConnell and Emily Chien at Dixon Middle School, Scott Bartholomew and Blake Johnson at Vista Heights Middle School, and Judson Birkel and Courtney Ward at Lehi High School, and Kiya Hall who was a great help as a consultant and put in countless hours as a grader. Particular thanks to Geoff Wright, who helped me piece this whole project together, and Peter Rich, and his extensive consultation in and out of class, particularly the help at AACE and AECT. My appreciation to the other TEE professors, Drs. Shumway, Terry, and Christensen. I need to thank my statistics team from Dr. Eggett's (The Egg) class, Jared Merrill, Katie Gustafsen, and Erica Hansen who helped put together the statistical analysis. A big thanks to my family for their patience while I've been so busy, "daddy can now play again." I am greatly appreciative of the love and support of my lovely wife Amber – first a compiler for undergrad, now a thesis. And finally, my appreciation to Emmanuel Schanzer, the creator of Bootstrap and who has been a very helpful consultant throughout this research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

This study evaluated the effects of participation in a course of Bootstrap on students' understanding of algebra, namely functions and variables. Bootstrap is a computer-programming curriculum for secondary age students that teaches programming video games through Racket, a functional programming language. While the curriculum is an introduction to programming using a gamification approach, the intent of the curriculum is to introduce students to algebra. To build a computer game, the students of Bootstrap use functions, function composition, and conditional (piecewise) functions. It was hypothesized that the students would transfer these concepts to algebra.

This study was conducted by (1) giving students of Bootstrap courses and corresponding control groups pre and post evaluations consisting of several algebra problems, and (2) interviewing students regarding their experience in the Bootstrap course. The differences of the post and pre evaluation scores were analyzed using multiple regression and effect size. The statistical results showed that students made a significant improvement in understanding variables, and suggestive improvement in understanding functions. It was demonstrated in the interviews that students could transfer the simpler concepts of functions from programming to algebraic notation, but struggled with the advanced concepts of composition and piecewise functions.

1.1 Statement of Problem

The 2008 President's National Mathematics Advisory Panel (U.S. Department of Education, 2008) stated,

The panel recommends that computer programming be considered as an effective tool… for developing specific mathematics concepts and applications, and mathematical problem-solving ability. (p. 52)

There has long existed a highly correlated relationship between academic success in mathematics and computer programming (Rich, Leatham, & Wright, 2012). Using computer programming to teach secondary school students the mathematical concepts of variables and functions has been researched since 1967 (Feurzeig, 2010). Despite years of research, the practice has not been generally adopted by secondary education (Johnson, 2000). Schanzer (2011) argues the reason for minimal adoption of algebra pedagogy through programming is because much of the curricula use imperative programming styles with languages such as Logo and BASIC. Contrariwise, Schanzer's Bootstrap curriculum uses the functional language Racket. The curriculum is an effort to help students with algebra while programming computer games (BootstrapWorld). But because of restrictions placed on the afterschool programs using Bootstrap, no formal studies have been conducted examining the value of the Bootstrap curriculum with regards to students' understanding of mathematics. This study researched both quantitatively and qualitatively how participation in a course of Bootstrap affected students' understanding of the algebraic principles of functions and variables, and the transfer from programming to algebra.

1.2 Background

Bootstrap is part of the outreach program TeachScheme! (Felleisen, 2010), which is a project that addresses the weakness of beginning computer programming education. The curriculum and tools developed from TeachScheme! address many of the issues of industrial languages, such as difficult syntax and complex development environments, that often repel secondary students away from studying computer science (Krishnamurthi, Felleisen, & Fisler, 1999). Further, the curriculum uses functional programming, which has the benefit over imparative languages of simple syntax and style (Schanzer, 2011). TeachScheme! was later expanded with the intent to become a basic part of secondary schools' core curriculum (Felleisen, Finder, Flatt, & Krishnamurthi, 2004a). By using functional instead of imparative programming, it was hoped that the TeachScheme! project would also help students with mathematics (Felleisen, 2011). The Bootstrap curriculum was later created as part of the TeachScheme! project, with the purpose of teaching math concepts through functional programming (Felleisen, 2010).

1.2.1 TeachScheme!

The TeachScheme! project began in 1995 in an effort to use research on functional programming to change K-12 computer programming curriculum (Felleisen, 2010). The concern was that existing curriculum based on current popular languages focused far too much on language syntax, referred to by Felleisen, Finder, Flatt, & Krishnamurthi (2004b) as "the tyranny of syntax" (p. 1). As Pea and Kurland (1984) explained, "The myth embodied in most programming instruction [is] that learning to program is 'learning facts' of programming language semantics and syntax" (p. 161). Instead, TeachScheme! researchers used a subset of the Scheme programming language to carefully introduce students to programming concepts without exposing students to unnecessary syntax or complex structures (Felleisen et al., 2004b). Further, TeachScheme! included a development environment called Dr. Scheme, which is a programming scaffold environment designed specifically for teaching computer programming (Findler et al., 2002).

TeachScheme! addresses the problem of teaching popular industrial programming languages to beginning programming students by using subsets of Scheme, presenting only what is necessary to the students as they learn the various programming concepts. As Felleisen (2010) explained,

> The "!" in *TeachScheme!* is a pun. One interpretation suggests that the goal of the project is to teach Scheme. It isn't, because the alternative explanation says that "!" is postfix notation for "not." While we never had the intention of teaching plain Scheme, lab observations during our first year drove home the important point that *no off-the-shelf programming language is suitable for novices*. (p. 130)

TeachScheme! is not intended only for computer science students. Felleisen (2010) argued that a "design-based programming curriculum can benefit everyone" (p. 129). Felleisen et al. (2004) further described programming as a creative process with a tight feedback loop. Bloch et al. stated,

> We are on a mission to turn computing and programming into an indispensable part of the liberal arts curriculum. Computing and programming teach skills just as fundamental as, and closely related to, essay writing in English and problem solving in mathematics. (2011, Overview)

Indeed, a mission of TeachScheme! was to aid in mathematics education. Felleisen (2001) explained how the intent was not only to help students with the design-oriented curriculum, but also to help students learn mathematics. But there has been little effort to present TeachScheme!

to the mathematics education community. Instead, the team has relied on Bootstrap as the outreach to mathematics (Felleisen, 2010).

1.2.2 Bootstrap

The Bootstrap program began as a PLT Scheme summer camp in 2005, sponsored by Northeastern University. The curriculum from the summer camp was later adapted for teaching at Citizen Schools, an afterschool program in Boston (Schanzer, personal communication, April 11, 2011). Bootstrap consists of nine lessons, each taking approximately 1½ hour. The instructors are at liberty to span some lessons across multiple class periods if needed. The goal of the course is to help each student write his or her own video game, similar to a game shown at the beginning of the course.

During the course, students are exposed to parameters and functions as defined in Racket. Additionally, students learn about the Cartesian coordinate system as part of placing objects on the screen. The Pythagorean Theorem is taught to help students find the distance between objects. Felleisen (2010) claimed, "*Bootstrap* provides the strongest evidence yet that teaching functional programming directly affects the mathematics skills and interests of K-12 students" (p. 130).

1.2.3 Racket

Racket is a language derived from the functional language Scheme. Guy Steele and Gerry Sussman created Scheme in 1975 as a specialized version of LISP (Steele, 2006). PLT Scheme was derived in 1995 as a language for beginning programming classes as well as introducing programming as a core subject in secondary education (Felleisen, 2011). PLT Scheme was renamed to Racket in 2010 (Racket).

Racket uses an algebraic functional notation. Functions are represented as prefix style, operator followed by parameters, delineated by parentheses. For example, a function to calculate addition of one and x is shown in (1-1):

```
(+ 1 x)                                        (1-1)
```

Beyond the example (1-1), only two other syntactical constructs are presented to students in the Bootstrap curriculum; the *define* and *cond* keywords. New functions are defined with the *define* keyword, as shown in (1-2):

```
(define (add_one x)                            (1-2)
        (+ x 1))
```

In example (1-2), add_one is the name of the function, and x is the name of a parameter passed to the function. *cond* is the keyword to create conditional functions. Conditional functions in Racket are similar to piecewise functions in algebra. With conditionals, a programmer is able to return different values based on the information given the function. An example is shown in (1-3).

```
(define (my_function a b)                      (1-3)
        (cond
                [(< a b)(− a b)]
                [else (− b a)] ))
```

The function my_function accepts two parameters, a and b. If a is less than b, then the function returns b subtracted from a. Otherwise, the function returns a subtracted from b.

Using Racket, students learn that algebra applies to more than just numbers. Felleisen & Krishnamurthi (2009) explained, Modern arithmetic and algebra do not have to be about numbers alone. They can just as well involve images, strings, symbols, Cartesian points, and other forms of "objects"… Algebraic expressions can both consume and compute pictorial values, enabling students to manipulate images using algebra. (p. 38)

Early in the Bootstrap curriculum students are taught functions to manipulate strings and images. In lesson 2, students are introduced to the string-length function and functions to draw shapes as shown in (1-6).

```
(circle 100 "solid" "red")                                        (1-4)
```

The example (1-4) indicates that a circle with the diameter of 100 should be drawn with the solid color red. In supplemental exercises, students are shown how to draw flags. The example below draws the Japanese flag using the function put-image, which places an image inside another image. The parameters are the inner image, coordinates of the inner image, and the outer image.

```
(put-image (circle 50 "solid" "red")                              (1-5)
        150 100
        (rectangle 300 200 "outline" "black"))
```

The code in (1-5) not only shows algebra manipulating images, but also demonstrates how composition of functions is a natural feature of the language.

1.2.4 Functional Programming

Advantages of functional programming over imperative programming in algebra pedagogy include the close relationship between the language and algebra in terms of syntax and conceptions. In imperative programming, programs consist of variables and procedures, sometimes called methods or functions. Variables in imperative languages do not represent a value. Instead they represent a storage bucket for values, and the contents of the bucket can change at any time (Schanzer, 2011, p. 42). Functions in imperative languages are very different from algebraic functions. A programming function in an imperative language can return a different value dependent upon input and state, or even define state, whereas by definition a function in algebra will always return the same value for the same input (Hinsen, 2009). Fortran, BASIC, C, C++, and Java are all imperative style programming languages.

"Functional programming, in its 'purest' sense, is rooted in how functions, variables, and values actually work in mathematics" (Wampler, 2011, p. 7). Functions are a mapping from input values to output values (Hinsen, 2009). They have an input list of zero or more parameters, a definition of the computational process, and a defined output (Feurzeig, 1986). The functions do not use variables in the imperative programming sense that is as storage buckets that can be altered at any time. In place of variables, functional programming uses parameters, though parameters are often referred to as variables. The parameters passed to the function do not vary; they cannot be changed within the function. The concept of assignment is outside the scope of Bootstrap's language. This is why functional programming functions do not have side effects. Functions cannot alter state, either inside or outside of themselves (Hinsen, 2009). Hinsen explained,

> If a program is composed of functions, and functions aren't supposed to change any variables, then what are variables good for? Nothing, and that's why functional programming doesn't have variables. (p. 87)

Variables in algebra are not the same concept as variables in imperative programming (Schanzer, 2011). In algebra, variables are given to a function as input. The function does not modify the value of the variable, or store its value for later use. Like functional programming, a function in algebra uses the variable to perform an operation. Felleisen et al. (2004a) stated, "A functional language is conceptually just a generalization of algebra" (p. 6).

An example of functional programming used in the Bootstrap curriculum involves the Pythagorean Theorem. A student is required to determine the distance between two objects on the screen. The X and Y coordinates of both objects are known. In math, the situation may be defined by the functions in (1-1) written in algebraic notation:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{1-6}$$

Using the programming language Racket, the problem may be defined as shown in (1-2):

```
(define (distance object1x object1y object2x object2y)                    (1-7)
     (sqrt (+ (sqr (- object2x object1x)) (sqr (- object2y object1y)))))
```

Both examples define a general function that suites the specific problem at hand. The difference between the two examples is primarily prefix notation and style.

Imperative languages do not share the same mathematic pedagogical properties as functional programming. Instead, imperative languages use variables that vary; they can be changed within the function. The following code sample is a typical way to write in Java the distance function using the Pythagorean Theorem.

```
double distance(int x, int y, int x1, int y1) {                           (1-8)
  double ans = 0 ;
  ans = ans + Math.pow(x - x1, 2);
  ans = ans + Math.pow(y - y1, 2);
  ans = Math.sqrt(cSquared);
  return ans;
}
```

Shown in (1-8) is a method that calculates the distance between two points using four steps. The result of each step is stored, storage bucket style, in variables using the assign (=) operator. In the imperative example, (1-8), the code describes how to find the distance. In the functional programming example (1-7), the function describes what the distance calculation is.

Subramaniam (2008) explained that functional programming is "a declarative style in which you say what to do instead of how something should be done" (p. 10).

1.3 Motivation for Study

Knuth, Alibali, McNeil, Winburg, and Stephens (2005) declared that algebra is considered to be a "gatekeeper" to future study and employment opportunities (p. 48). Krishnamurthi et al. (1999) also explained that algebra skills are important "for students' professional advancement," regardless of the field of employment (p. 13). But there are common misconceptions among students in the categories of functions and variables. A common misconception is to think of a variable as something to be solved, i.e., solve for x (Schanzer, 2011). Feurzeig et al. (1970) explained that sometimes it represents a number, or something called a variable, or a function (p. 14-15). Few students grasp the true nature of variables, which represent a range of numbers at once (Knuth et al. 2005). Functions are also presented in mathematics curricula in a confusing manner. Simply defined, a function is a mapping from an input to an output (Hinsen 2009). But to students, functions are presented many different ways, which initially do not seem related. Functions are presented as: a domain and range, a mapping between sets, key value pairs in a table, graphs on a Cartesian plane, and as equations such as $f(x)=5x+2$ (Schanzer, 2011). Regarding the current pedagogical practices concerning functions, Schanzer (Ibid.) stated the following,

> Unfortunately, each representation brings its own jargon, skills and properties along with it. Students spend months learning how to graph functions, and years learning how to play with equations. We show all these representations to students and do our best to point out the connections between them and teach the larger "*function*" concept. (p. 2)

In July 2010, Conrad Wolfram presented a TED (Technology, Entertainment, Design) talk about "his radical idea: teaching kids math through computer programming" (Exploring Computational Thinking). Actually, the idea has been researched since 1969, with the release of Feurzeig et al. (1970) paper to the NSF (National Science Foundation). In that paper, understanding the abstract concepts of variables and functions were highlighted as an advantage to teach mathematics through programming. The paper also stated how programming could be a laboratory for mathematics.

> How could a person set in motion a sequence of mathematical events or a mathematical process unfold? Using a computer with an appropriate programming language adds this extra dimension to mathematical experience. (p. 17)

Noss (1986) explained further,

> If algebraic abstraction, in the sense of symbolic representations of relationships, is central to mathematics, then the twin ideas of function and variable are central to algebra. Yet it is precisely these ideas which children find such a major stumbling block in their learning. (p. 336)

The strong correlation between the two disciplines of mathematics and computer programming has driven studies and implementations for decades. Since the Feurzeig et al. (1970) report in 1969, significant changes in many areas have been made. Computer programming languages have matured, the microcomputer was invented, leading to the proliferation of the personal computer, and graphical user interfaces have become commonplace. Programming development practices have also improved. But using programming to supplement mathematics education has still failed to be used widely. As Johnson (2000) stated,

> I would suggest here that the research and development activities over the past 20 or 30 hears has failed to find a 'home' in the curriculum for mathematical programming, i.e., researchers and developers have operated as if they expected the programming environments themselves would become the curriculum… I do not propose to pass judgment on such a position, other than to note that it would seem to have failed to gain support on any large scale. (p. 202)

In light of the many benefits of functional programming, and because past research investigating correlations between mathematics and programming have primarily considered imperative programming, it is worthy of research to examine how functional programming can be used in algebraic pedagogy.

Bootstrap has strong potential to enhance efforts to help students better understand algebra. Felleisen and Krishnamurthi (2009) asserted, “Any attempt to align programming with mathematics will fail unless the programming language is as close to school mathematics as possible” (p. 39). Bootstrap uses the well-designed functional language Racket, one that can be understood by the general student population (Felleisen 2010). Students are introduced to the programming language on a just-in-time basis to avoid confusion.

1.4 Purpose

The hypothesis of this study was that students who participate in a Bootstrap course would have gained a better understanding of algebra at the end of the course than their equivalent peers who were not in the course. Schanzer had not been able to conduct formal studies due to the restrictions of the afterschool programs that he has taught. Schanzer stated that studies from 18 Bootstrap courses would be enough for him to statistically demonstrate the value of the Bootstrap curriculum. This study was the first to contribute formally analyzed courses. In addition, the findings from this research project (1) assisted a BYU research team investigating the value of teaching programming to K-9 students to improve mathematical self-efficacy; and (2) contributed to the understanding of how programming and mathematics can complement each other in educational settings.

2 LITERATURE REVIEW

Using computer programming as a tool for algebraic education has been researched for four decades. Studies began with the Logo programming language (Feurzeig, 2011; Milner, 1973), and later with Logo turtles (Feurzeig, 2010). Some research exists in studying other languages, such as BASIC (Boyes et al., 1985) and Scheme (Paz & Leron, 2009; North, 2005). But despite the years of research, the practice has not been adopted on any wide scale (Johnson, 2000). Further, as computers have become widely available with powerful graphical interfaces, programming as part of mathematics education has become scarce while mathematical activity programs have become popular (Robertson, 1998; Feurzeig, 2010).

2.1 Logo

Logo is an interactive programming language used primarily for K-12 and post-secondary education. Feurzeig (2010) explained the purpose of Logo.

> It was specifically designed to be a powerfully expressive yet readily accessible programming language for construction, exploration, and investigation of ideas and processess in mathematics, science, language, and music while providing children a lively learning environment. (p. 257)

There are several important similarities between Racket and Logo in the context of the Bootstrap curriculum, and Logo. Both languages were designed for pedagogical purposes, as opposed to industrial uses (BootstrapWorld; Feurzig, 2010). They both have interactive environments

(Feurzeig, 2010; Krishnamurthi et al., 1999; racket-lang.org). Both have environments which present functionality on a need-to-know basis (Howe, O'Shea, and Plane, 1980; Bloch, 2007). Both are derivatives of the Lisp language (Steele, 2006; Feurzeig, 2010). Finally, both the Bootstrap curriculum and Logo were designed as a pedagogical tool for algebra (BootstrapWorld; Feurzig et al., 1970). But a key difference between Bootstrap with Racket and Logo is that Logo can use an imperative form of programming, and is used imperatively in Logo Turtles, while programming in the Bootstrap curriculum is purely functional (Pea and Kurland, 1983; Schanzer, 2011). Logo Turtles also uses a constructionist teaching model, inspired by the work of Jean Piagnet (Papert, 1993), while Bootstrap curriculum follows a direct instruction approach. The history of Logo gives context to the language's structure, purpose, success and shortcoming, and an understanding on how the language compares to Racket and the Bootstrap curriculum.

2.1.1 Background

Feurzeig (2010) explained that in the mid 1960's, two advances in computer technology led to the creation of the Logo Language. The first technological advancement was computer time-sharing, which made remote computer stations to schools a possibility. The other advancement was the development of "conversational" or interactive programming languages. Regarding early interactive languages Feurzeig explained,

> I saw dramatic new possibilities for such programming languages in education. My interest shifted from earlier work on tutorial systems to the development of interactive programming languages for children, languages specifically designed for learning. Initially we focused on making mathematics more accessible and compelling to beginning students. (p. 259)

In 1965-66 Feurzeig, with several other collaborators including Papert, introduced programming as a tool to teach mathematics at schools that were part of the time-sharing system at BBN (Bolt Beranek and Newman, a Cambrige research firm). Feurzeig explained that using a high level language to learn mathematics highly motivated students, but the language used (TELCOMP) was inappropriate for educational purposes. The language was focused on computation, and "generally lacked facilities for nonnumeric manipulation" (Feurzeig 2011, p. 2).

> These languages also had serious deficiencies in control structure, e.g. lack of support for modularity and recursion. Many, such as the recently developed BASIC language, lacked procedural constructs. Most had no facilities for dynamic definition and execution. Few had well-developed and articulate debugging, diagnostic, and editing facilities, essential for education applications. These considerations led to the development of Logo. (Feurzeig, 2010, p. 259)

Logo was subsequently designed with the following requirements,

> 1. Third-graders should be able, with very little preparation, to use it for simple tasks.
> 2. Its structure should embody mathematically important concepts with minimal interference from programming conventions.
> 3. It should permit the expression of mathematically rich, non-numerical algorithms, as well as numerical ones. (Ibid., pp. 259-260)

Logo was derived from the language Lisp, keeping much of the same computational power of Lisp, with syntax that "is more familiar and accessible" (Feurzeig, 2011, p. 3**)**. The name Logo came from the Greek word Logos, "the word or form which expresses a thought, or thought itself" (Feurzeig, 2010. p. 260).

With support from the US Office of Naval Research, Logo was pilot-tested in the summer of 1967 with fifth- and sixth-grade math students (Ibid). In 1969, Feurzeig, Papert, Bloom, Grant, and Solomon submitted their report to the National Science Foundation on research using 7th grade students (Feurzeig et al., 1970). The report gave some observational discussion on why programming should be used "as a foundation for an integrated course in mathematics" (p. 17). The report also made the following statement concerning functions and variables,

> Programming can be used to give students very <u>specific</u> insights into a number of key concepts. Ideas such as variable and function remain, to say the least, obscure for many high school students. Indeed, college students often have trouble with the many roles of the "x" in algebra: sometimes it appears to be a <u>number</u>, sometimes a subtly different kind of object called a <u>variable</u>, and on other occasions it is to be treated as a <u>function</u>. We contend that the difficulty stems less from the intrinsic intellectual subtlety or complexity of these distinctions than from their ethereal relation to anything in the real and familiar world… In programming, the distinctions arise concretely; they must be faced; and the physical nature of the machine provides a more earthy reference than can any abstract work. (pp. 14-15)

Some of the early programs were designed to turn English sentences into Pig Latin, play strategy games, and to generate concrete poetry (Papert, 1993, p. 281).

In 1970, Papert founded the Logo Labatory at M.I.T. His purpose was to create a new education system, based on his earlier work with Jean Piaget (Papert, 2000). Papert wrote,

> In many schools today, the phrase "computer-aided instruction" means making the computer teach the child. One might say the *computer is being used to program* the child. In my vision, *the child programs the computer* and in doing so, both acquires a sense of mastery over a piece of the most modern and powerful technology and establishes an intimate contact with some of the deepest ideas from science, from mathematics, and from the art of intellectual model building. (1993, p. 5)

Papert used the Logo language to create an educational tool called Logo Turtles. The turtle initially began in 1971 as a floor robot, controlled by Logo commands (Papert & Solomon, 1971; Feurzeig 2010). Screen turtles were later developed in 1972. Using turtles, it was shown that elementary students could program (Papert & Solomon 1971). Papert felt that Logo Turtles was "the proper way to introduce <u>everyone</u> of whatever age and whatever level of academic performance, to programming" (Papert and Solomon, 1971).

Feurzeig (1986) continued of the use of Logo as an educational tool, and developed "a Logo-based introductory algebra course for sixth graders" (p. 229). The goal was to develop programs that may be interesting to students, such as knock-knock jokes and secret codes. Feurzeig stated, “These will lead in a natural way to projects that address the standard content of introductory algebra” (p. 230). Feurzeig also stated that he chose Logo because of its “essential mathematical character (e.g. its recursive functional structures which account for its power and elegance” (Ibid.).

Logo’s influence has a strong presence in the topic of pedagogical mathematics through programming, and generally is a standard by which the efficacy of other languages is compared. In 2008, the President’s National Mathematics Advisory Council report stated the following.

> Effects are larger if the computer programming language is designed for learning (e.g., Logo) and if students’ programming is carefully guided by teachers so as to explicitly teach students to achieve specific mathematical goals. (p. 52)

Regarding computer based instruction, Logo is the only proper noun in the entire report. Logo is indicated as the language “designed for learning.”

2.1.2 Logo and Bootstrap Compared

Logo and Racket have some very important similarities and differences. They are similar in that they both were designed for pedagogical purposes, particularly in mathematics. Both also claim to use a language that is “expressed in standard formal algebraic notation” (Feurzeig, 1986, p. 251). Both are derived from Lisp. Pea (1983) used the following adjectives concerning Logo,

> The elegance and beauty of Logo [is] that [it] derives from its parent language, LISP, used in artificial intelligence, its procedurality which allows one to define new procedures and use them as building blocks in increasingly complex programs, its control structures that allow very brief recursive programs that can solve quite difficult problems, [and] the use of conditional tests. (p. 31)

Harvey (1985) called Logo “by far the most powerful programming language available for home computers” (p. xi). A primary difference between Logo and the Bootstrap curriculum is shown within their pedagogical intent. The Bootstrap curriculum has the purpose to "explicitly teach students to achieve specific mathematical goals" (BootstrapWorld). Meanwhile Logo presents a vast set of possible but not enumerated benefits. The Bootstrap curriculum better meets the standard suggested by the 2008 President's National Mathematics Advisory Council report.

> However, computer programming by students can be employed in a wide variety of situations using distinct pedagogies, not all of which may be effective. Therefore, the findings are limited to the careful, targeted application of computer programming for learning used in the studies reviewed (p. 51).

Bootstrap endeavors to present algebra using Racket's algebraic structure. A key part of that structure is that the language requires functional programming. Logo is used in an imperative manner. Instructions are given step-by-step, explaining how rather than what should be done. Concerning Logo, Schanzer (2011) stated,

> Despite Logo's relationship with the LISP programming language, the programming model exposed by Logo is far from functional. An algebraic construction of animation, for example, might view position as a function of time, whereas a Logo procedure would use repeated "MOVE" statements. (p. 48)

Instead of using functions, the Logo language uses procedures. Harvey (1985) stated, "A *procedure* is like a recipe or a technique for carrying out a certain kind of task" (p. 12). The task mentality is foreign to functional programming. Hinsen (2009) explained, "A mathematical function doesn't 'do' anything other than return a value" (p.87).

Other imperative aspects of Logo are the use of variables and assignment. Logo uses variables in a way that contradicts the definition of functional programing, that is, variables are declared within (and without) procedures and assigned values. Logo uses the MAKE keyword as an assignment operator. The assignment changes the value of a given variable, as code example (2-1) shows.

```
MAKE "NEW :OLD                                                    (2-1)
```

Harvey (1985) explained that the first input to MAKE is the name of the variable, while the second input NEW is the value to assign to the variable OLD (p. 49). In Logo, variables have local and global scope. A variable with global scope is one that can be modified by any procedure in the program, which by definition is a side effect. There are functional aspects of Logo, such as the use of recursion, but these are outside the scope of this study. Instead, this study focuses on programming used to teach middle school and high school age student to the algebraic concept of variables and functions.

Much of the research about Logo is more explanatory than research. Papert explained that this is by design,

> In Mindstorms I made the claim that children … learning to program can affect the way they learn everything else. It did not occur to me that anyone could possibly take my statement to mean that learning to program would *in itself* have consequences for how children learn and think. (2000, p. 727)

Papert was amazed to see experiments done and papers written "on 'the effects of programming (or of Logo or of the computer)' as if we were talking about the effects of a medical treatment" (pp. 727-728). Regarding this example Papert clarified,

> The difference between these two conceptions of the role of programming is of the same kind as the difference between the two interpretations of Piaget: in both cases the crucial difference is between primacy of the epistemological (talking about ideas) and primacy of the psychological (talking about how a person is affected by a treatment) (p. 728).

Pea (1983) disagreed with Papert, as explained below,

> While Papert and colleagues undertook extensive studies of the children doing Logo programming in Brookline school system, their reports of this work were principally qualitative in nature, citing and discussing some of the programs that were created by the

> children, the global differences in programming style that seemed to be intuitively distinguishable… Though interesting, these reports do not directly address the widely touted claims for the development of thinking skills that transcend the programming context, for which case-study methods are inappropriate (p. 27).

Given that this study is based on evaluating the value of the treatment, participating in a course of Bootstrap, it is valuable here to examine the "primacy of the psychological" done in the past. But rather than trying to give a summary of research on the Logo language as a whole, this review focuses on points comparable to the intent of the Bootstrap curriculum; namely the benefit of programming to middle school students in learning the algebraic concepts of variables and functions. Described here are examples of studies that are comparable to the Bootstrap courses examined in this study in terms of the age of the participants and the general expected benefit of the treatment.

2.1.3 Studies

An early study by Milner (1973) consisted of teaching Logo to 18 fifth grade students randomly selected from students at Oakleaf School in rural Pittsburgh. Twenty other randomly selected students were assigned as a control group. Students worked in groups of 4 on a time-sharing DEC System-10 with each student have their own teletype device, KSR-33. The study consisted of 3 phases. In phase I students were taught the Logo programming language twice a week for 40 minutes. Some of the lessons came from the BBN. In phase II, students were grouped into levels of high and low ability, based on their scores from Stanford Achievement Test. Then they were randomly assigned into one of three instruction groups. Milner explained,

> One method consisted of an algorithm given in natural language form to be programmed in the LOGO language by the students. The algorithm was based on a task which was also given to the student. In the second method, students were given an incomplete computer program written in the LOGO language. It was necessary for the students to

complete the program, which was also based on a task given, and implement it on the computer. In the third instructional method, students were given no information other than the task definition. (p. 7)

In the third phase, students were given similar tasks to the previous phase, "except that no explicit information was given to them other than the task definition" (Ibid.). Pre and post tests were administered to the treatment and control groups. It was found from the statistical analysis on those tests that the instructional method and ability were insignificant. The analysis also showed that the students did gain a significant better understanding of variables from Logo programming instruction.

A study by Howe et al. (1980) showed that a class of 11-13 year old boys who struggled with mathematics improved their understanding of the subject after two years of studying programming Logo. Their stated objectives included improving "the boys' ability to handle particular topics in the school mathematics curriculum, such as the use of the function to express the relationship between two sets of numbers (p. 92). Students were taught programming for one hour a week, during class time the first year and after regular school hours the second year. The first year was dedicated to teaching programming, the second year they used programming to explore the mathematical concepts they had trouble with. A separate class with similar mathematical abilities was used as a control group. The metric used in the study was the students' "standing" among peers with regard to mathematics. About half of the boys in the experimental group improved their standing, with none dropping to a lower rank. Meanwhile in the control group, one boy improved his standing while one boy dropped. Furthermore, the teachers commented that the boys in the experimental group demonstrated that they "could argue sensibly about mathematical issues" and "explain mathematical difficulties clearly" (Howe, Ross, Johnson, Plane, and Inglis, 1982). The boys in the control group did not demonstrate this capability. Regarding this assessment, Pea and Kirland (1984) stated that “the reliability of such ratings is questionable, since the math teachers should have been blind to which students learn Logo” (p. 159).

Pea (1983) described multiple studies on Logo done at the Center for Children and Technology, Bank Street College of Education. In one study, children were given a three part written assessment consisting of Logo command understanding, writing Logo programs, and finding error in programs. In the writing program section, it was found that students could write programs with a series of instructions, but “many children could not write a version of such a program using a variable” (p. 28). A second study “utilized a series of increasing complex logo programs to reveal the depth of understand … in a half dozen of the best programmers in the two Logo classrooms” (Ibid.). Though the study was focusing on recursion, an additional observation was that “the students often displayed production without comprehension” (p. 29). The students would use variables and test statements in one program, but not show understanding in another program. Pea explains that “rote use of ‘chunks’ from other children’s programs or those of the teacher seems to be responsible for this rigidity of use” (Ibid.). One of Pea’s conclusions is “the transfer of problem-solving strategies between dissimilar problems, or problems of different context, is notoriously difficult to achieve even for adults” (Ibid). Pea’s conclusion was that Logo still has potential and should not be disregarded, but “with thoughtful instruction … we expect that Logo may provide a good window for the child into these important computational concepts” (p. 31).

The aim of the 1986 Noss study was “to examine the kinds of thinking which children who had learned Logo for 18 months (approximately 50 hours), could carry over to an algebraic context" (p. 339). The students were measured by if they could "(a) construct meaningful symbolizations for the concept of variable (b) construct formalized (algebraic) rules" (p. 340). The end study consisted of interviews of eight children. Noss took the students through a series of questions involving variables, similar to what would be learned in Logo programming. The results were mixed. "The interpretation of the data offered here… is that children may – under the appropriate conditions – make use of the algebra they have used in a Logo environment, in order to construct algebraic meaning in a non-computational context" (p. 354). The author concluded that Logo programming alone is not sufficient for education, but needs "pedagogical input" (p. 334). Schanzer (2011) explained that a problem with the Noss study is that a control group was not used. The study "does not address students would actually perform better than their non-programming peers when confronted with the same problem" (p. 47).

In Sutherland's (1989) research it was concluded that Logo does help students better understand the algebraic concept of variable, but that understanding is dependent upon "the nature and extent of their Logo experience" (p. 341). An aim of the study "was to develop and evaluate materials to help pupils make the links between variable in Logo and variable in 'papers and pencil' algebra" (p. 321). The study consisted of three years of qualitative research by video recordings and observations with eight 11-14 year old students. At the end of the study, structured interviews regarding variables were given to each student. All the students recognized that a variable represents a range in Logo, but some did not transfer this knowledge to algebra. Further, while some students understood that different names could represent the same value, all but one did not recognize the same concept in algebra.

In 1991, Ortiz and MacGregor studied sixth-grade students (n=89) on the effectiveness of Logo in understanding variables. Students from four classrooms in two metropolitan public schools participated in the study. All students were given four pre-tests to “provide measures of mathematics achievements, logical thinking ability, understanding of the concept of variable, and attitude towards various aspects of learning (p. 39). Three of the tests were the *Comprehensive Tests of Basic Skills* (Level H, Form V), *Test of Logical Thinking*, and *Robustness Semantic Differential (RSD)*. The results of those three tests given as pretests were analyzed and showed no significant difference between the later experimental and control groups (p. 44). The forth test was *Understanding of the Concept of Variable Instrument (UCVI)*, developed by the investigator (p. 41). All students received five lessons on the use of the Logo programming language, but were not taught about variables. After the five lessons, students were randomly divided into three groups, two experimental groups and one control group. One experimental group learned about variables using programming, and the other experiment group learned about variables using textbooks. Both experimental groups received five fifty-minute lessons. The control group did not receive any instruction on variables. The RSD and UCVI were administered to all students at the end of the treatment. The UCVI was administered again after three weeks to test retention. Students taught variables using Logo programming showed a better understanding than students without instruction on variables ($p < .01$), but there was no significant difference between students taught variables through Logo and students taught variables through textbooks. But on the retention test, the Logo group scored significantly higher ($p < .01$) than both the textbook and control groups. “In fact, Logo students’ score increased from the immediate to the delayed posttest, while the scores of the students in the textbook group declined” (p. 47).

Review articles by Clements (1985) and Clements and Sarama (1996) offer a summary of Logo research from the two decades. The first review article considered the following pessimistic view, “Because children’s concepts of Logo – including such powerful ideas a as variable, recursion, modularity, and so on – are limited, the turtle might need to be dropped from the race" (1985, p. 69). The studies by Milner (1973), Howe et al. (1980), and Pea (1983) are referenced, along with a 1984 Noss study in which students "are able to make use of Logo knowledge to make algebraic generalizations in a non-Logo context" (p. 62). The conclusion of the Noss study is that while the reports were conflicting, the following pattern was observed, "Most reports of success observe children within the context of Logo programming, without measuring transfer" (p. 62). The conclusion is to not abandon Logo and Turtles, but "slow and steady – that is, cautious and thoughtful – is probably the best way to attain the good" (p. 69).

The 1996 Clements and Sarama review article spoke highly of the Logo language and its pedagogical value, and begins with a strong review when it comes to the research on the value of Logo regarding variables and algebra. The authors quoted McCoy as follows, "Logo programming … is clearly an effective medium for providing mathematical experiences … This is particularly true in geometry … and the concept of variables" (p. 10). Clements and Sarama later stated,

> Our earlier review suggested that Logo experience would facilitate student learning of "generalized arithmetic" – variables and algebra. Research since then has supported this view. Logo can help students from primary grade to high school understand variables, even in comparison to other treatments. (p. 16)

But other studies reviewed in the Clements and Sarama article suggested the benefits of Logo might have been overstated. Lehrer and Smith showed that "some students do not fully generalize the variable idea as used in Logo to other situations" (p. 16). Referring to Boulay, the authors explained that students have troubles using variables in programming Logo. Students declare and then not use a variable in a procedure, or are confused as to what the variable stands for. "They over generalize analogies; for example, taught to think of variables as a box, many believe that a variable might hold more than one value" (Ibid.). Relying on the work of Noss and Hoyle, Clements and Sarama explained that while there does not seem that students have specifically come to understand variables and algebra, they might be building a conceptual framework that can be used later in learning algebra. Clements and Sarama summarized their review article by stating the following,

> There is some evidence that Logo provides an "entry" to the use of the power tool of algebra. It is an environment in which some students perceive the use of formalizations such as variables as natural and useful. Again, however, we find that students' ability to generalize their Logo-based notion of variable may depend to a great degree on the depth of their Logo experience and the instructional support given them. (p. 17)

2.2 Scheme

Studies using a functional language have been done more recently. North (2005), a high school computer science teacher in a Houston area school, in coordination with Rice University, conducted an action research study. Regarding using TeachScheme! she stated, "In my computer science classes I found I was often required to re-teach algebra concepts. By using programming to teach these algebra concepts, I have observed students over and over again finally saying 'I get it'" (North, 2005, p. 115). North's informal results were measured by success in district and state tests.

> Out of my experimental group, 100% passed the HISD (Houston Internal School District) Snapshot tests and 100% passed the state accountability ninth grade math TAKS (Texas Assessment of Knowledge and Skills) Test. This is compared to approximately 60% of the students at Westside High School passing the accountability tests. (p. 116)

Paz and Leron (2009), as educators in northern Israel, noted that in algebra the function was one of the most difficult concepts for students to grasp. But as they adopted Scheme in their programming classes, they found students were able understand functions better than before the course.

> This article is about mathematical issues that came up serendipitously during a functional programming course and not about the educational use of programming to help students learn mathematical functions. (p. 26)

The study consisted of 5 eleventh grade classes of about 20 students each, and 5 other classes from three other schools in the region. The students studied functional programming for about 3 weekly hours for the entire school year, totaling about 90 hours (p. 27). The study was purely qualitative, and the data was collected through observations and interviews. The study showed the courses did "serve as an effective intuitive support for learning about functions, but also that it can later clash with the formal function concept, leading to some persistent obstacles in understanding functions" (p. 37).

2.3 Recent Trends

Since the mid 90's, research into the theory of learning math through programming dropped significantly, which is in contrast to the substantial increase in power and availability of computers since that time. Initial Logo programs were simple word games, and subsequently drawing shapes on the terminal. Now there currently exist many curricula that allow students across a diverse age and skill set to create highly graphical interactive programs. Examples include the programming languages and environments of Alice, Scratch, Bootstrap, the recently created programming tools on Khan Academy, and Logo Turtles, used both with Lego Robots as well as patterns on the screen. Nevertheless, rather than having students create programs to teach mathematics, educational trends favored using computer programs that can be used to learn and teach mathematics; e.g. Math Blaster, Fast Math, Reflex Math Fact Fluency, Timez Attack, Coolmath.com, and multiplication.com.

Regarding the use of computer programming in schools in South-West Scotland in 1984 to 1988, Robertson (1998) explained,

> These ideas were based on the central precept that children may learn to control computers and that, in so doing, they may extend their learning beyond that which is typically possible without such devices. By 1997, it has become clear that these radical ideas were not sufficiently embedded as to survive the effects of later technological advances or of political reorganisation. (p. 31)

Johnson (2000) stated the following concerning the use of computer programming to teach mathematics,

> I would suggest here the research and development activities over the past 20 or 30 years has failed to find a 'home' in the curriculum for mathematical programming. (p. 202)

Johnson suggested two reasons for this. First is the lack of convincing empirical evidence of benefit. Johnson further stated,

> The result of such a limited practice is that programming becomes just another new topic for teaching and learning to be 'squeezed' into an already 'full' curriculum with little or no apparent future 'pay-off'. (Ibid.)

Johnson's second reason deals with the progression of computer technology. As computers were becoming commonplace in schools, they featured graphical interfaces and powerful applications. Johnson (2000) explained how this affects computer-programming pedagogy,

> Pupils in 'computer rich environments' now have access to a range of software packages – spreadsheets, databases, computer algebra systems (CAS), geometrics supposers, and modeling and simulation tools. Each of these packages also has its own 'overheads' in terms of teach and pupil time and effort to learn the basic elements for using and, the difficult part, applying, and finally an appreciation of the pedagogic implications for teaching and learning in such an environment. (p. 203)

The teaching of business applications had crowded out the pedagogical tool of teaching of programming. As Feurzeig (2010) laments,

> Computers are now nearly ubiquitous in schools throughout the US. They are used extensively for word processing and information retrieval. Instructional applications abound, often enhanced by visually rich graphics and animation. Because of the dramatic

> rate of development and application of computers, one might have predicted that the new learning experiences made possible by programming ideas and activities would be well established throughout schools by now. But, sadly, the use of high-level programming languages for student design and invention, particularly during children's formative years, has almost vanished. Their powerful potential as expressive tools for knowledge construction has yet to be realized. (pp. 264-265)

Schools are recognizing the value of computers as a source for interactive educational games. But as Kafai (2006) states, "Far fewer people have sought to turn the tables: making games for learning instead of playing games for learning" (p. 37).

2.4 Discussion

Researching into the pedagogical value of programming for mathematics initially followed the history of the development of programming languages and environments. It was not until interactive programming environments became available that Feurzeig began to consider a language like Logo. And said environment depended upon the availability of a reasonable input mechanism, such as teletype machines. As computers were beginning to make their significant mark on society, the promises of the Logo language and paradigm presented by Papert naturally fit into the new environment. The following decades were a time of experimentation and validation. Even though some value was demonstrated, the evidence for the paradigm was not solid enough to survive the onset of inexpensive computers with graphical interfaces. Indeed, as graphical interfaces arose, computers as a pedagogical tool became less about programming and more about using applications.

It is interesting to note that Scheme did not appear on the research scene as a algebraic education tool until 1995, and even then indirectly. Felleisen commented early on that Scheme may be helpful in understanding algebra, but the notion was not implemented until 2005.

Meanwhile, both North and Paz and Leron demonstrated the potential of how learning Scheme may improve algebraic skills. But why didn't Scheme catch on earlier? The language has been around since 1975. It may have been the popularity of Logo, with its claims to be mathematical in notation, which kept Scheme from being examined for mathematical pedagogical value. Now that Schanzer has provided a curriculum that combines the power of today's graphical interface with the algebraic notational language of Scheme, new opportunities in the benefit of Scheme as an algebraic pedagogical tool deserve to be examined.

3 METHODS AND PROCEDURES

Three different groups of students using Bootstrap at different schools were studied. Treatment and control groups were given pre and post tests on algebraic concepts (Appendix A). The tests were analyzed using multiple regression and SMD (Standard Mean Difference) effect size statistics. The validation of the hypotheses was dependent upon statistical measurements of the data showing significant difference between the assessment scores of the experimental and control groups. Also, a sample of students was interviewed concerning the transfer of algebraic concepts learned from programming to problems written in algebraic notation.

3.1 Courses

This study evaluated three courses using the Bootstrap curriculum, all taught at schools local to BYU. The Bootstrap research team at Brigham Young University (BYU) coordinated the Bootstrap courses. The research team was responsible for arranging the classes with the schools, acquiring appropriate permissions from BYU, the school district, the school administrators, and the parents of the students. BYU students taught the courses. Each course had two instructors.

The first course was taught at Dixon Middle School (DMS) in the Provo, Utah School District. Dixon consists of students from grades 7 and 8. The courses were taught as an after school class where students volunteered to participate in the course. The duration of the course was 6 weeks, with 1 ½ hour sessions twice a week. The course was extended beyond six weeks to accommodate some students who started the course late. Only observational data was collected from the course.

The second course was at Vista Heights Middle School (VHMS) in the Alpine, Utah School District. The school teaches grades 7 and 8. This course was also taught as an after school class where students volunteered to participate. The course began with 14 students, with 9 completing. This course was the first to participate in the quantitative research with a control group. The control group consisted of 17 students from a CTE (Careers, Technology, Engineering) class taught during regular school hours. The course was taught for just over six weeks, twice a week. Each class lasted about 1½ hour.

The third course was taught at Lehi High School (LHS), also part of the Alpine, Utah School District. The course was taught as part of a video game programming class. Instead of an after school class, it was taught during the regular school day, and students received school credit for the class. It was taught every other day for 1½ hour for 8 weeks, after-which the class moved on to another programming curriculum. The students for the course were selected by the school administration as students struggling in math. Seven students completed the course. The control group consisted of 9 students from an algebra class.

The Bootstrap curriculum has very detailed lesson plans. The instructors of the courses received a few hours of training on the curriculum, but it was expected that the instructors would pick up the concepts by experimenting with the exercises themselves. The observational data for this study occurred at the beginning and the end of each course, at the time assessments were given and interviews were performed. No observations were made concerning the style or approach each teacher, nor any specific content outside the Bootstrap curriculum that may have been presented.

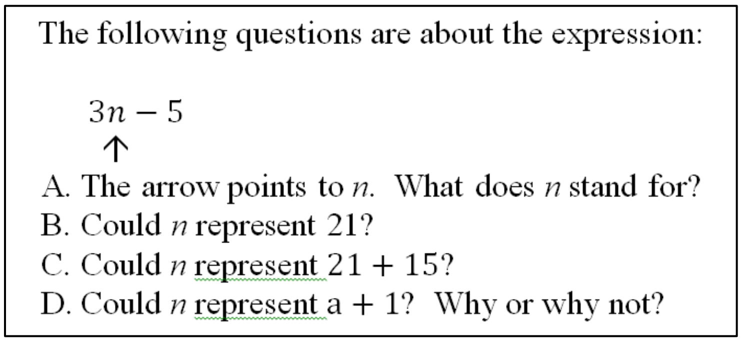
The instructors had varied programming background. One instructor at VHMS was a graduate student in Computer Science, and the two instructors at LHS were graduate students from the Instructional Psychology and Technology program at BYU. Two instructors at DMS and one instructor at VHMS were students of the Technology and Engineering Education program.

3.2 Assessments

Students in both the treatment and control groups were given pre and post math assessments. The assessments consisted of demographic questions and a series of math problems. The math problems were based on questions from: (1) Knuth et al. (2005) work examining students' understanding of core algebraic concepts, and (2) Concepts from the Bootstrap course as identified by the BYU research team. A member of the BYU research team scored the assessments using the rubric found in appendix B. In scoring the assessments, the questions were divided into the following five categories: variables, order-of-operations, functions, transfer of functional concepts from Bootstrap to algebra, and other algebra questions. Some of the questions in the functions category were also included in the transfer category. The point of the duplication between the functions and transfer categories was to gather as much information as possible about students' understanding of functions, as well as to document the students' transfer of specific concepts from Bootstrap to algebra. Those concepts were function composition, piecewise functions, Cartesian Coordinates, and the Pythagorean Theorem.

3.2.1 Questions on Variables

The assessments contained two questions on variables; both were based on Knuth et al. (2005) research. The first question on variables is shown in Figure 3-1.


The following questions are about the expression:

$3n - 5$
$\uparrow$

A. The arrow points to n. What does n stand for?
B. Could n represent 21?
C. Could n represent $21 + 15$?
D. Could n represent $a + 1$? Why or why not?

Figure 3-1 Variables Question 1

The purpose of the problem was to identify to what extent a student understands variables. The question as written was an expansion of the question presented by Knuth et al., where parts B, C, and D were added. It was in the interest of the study to find if the student understood that a variable could represent an expression, a vital concept in understanding function composition. In part A of the question, the student was required to first identify what n is. Parts B, C, and D then challenged the students' experience with variables, leading from n representing a number to representing an expression. The question was scored with 3 points possible. According to the rubric, no score was given for part A. The students received 1 point for answering "yes" on part B, and ½ point answer "yes" on parts C and D. One-half point was also awarded for each parts C and D for a correct elaboration. A common answer for parts C and D that were given full credit usually indicated that a variable could be anything. Other correct elaborations for part D included the following: “a variable can have a variable,” “yes, but you would have to solve for n,” and “(n+5) is data.”

Figure 3-2 shows the second problem on variables.

Which is larger, 2n or n+2? Please explain your answer.

Figure 3-2 Variables Question 2

The point of the question was to find if the student understood that a variable represents a range, as Knuth et al. (Ibid.) explained.

> [The problem] was designed to assess students’ abilities to use the concept of variable to make a judgment about two varying quantities. In particular, to be successful, students must recognize that the values of [2n] and [n+2] are dynamic and depend on the value of n, that is, they must view n as a variable – a literal symbol that represents, at once, a range of numbers. (p. 70)

The question was also scored on a three-point scale. The student received 1 point for the question if they answer 2n or n+2, accompanied by a correct example. The student was given 2 points if the student indicated the answer depends on the value of n, and gave correct examples. To earn three points, the student needed to give three examples shown n less than, greater than, and equal to 2. No points were given in the case of incorrect examples or indicating 2n or n + 2 were always greater than the other.

3.2.2 Question on Order-of-Operations

Unlike programming in other languages, order-of-operations does not apply to programming in Bootstrap’s language. In many popular languages, a series of operators and operands can be written with or without parentheses. If parentheses are not present, then the language will use order-of-operations to evaluate expression. As an example, the Groovy code in (3-1) is interpreted by performing the multiplication and division first, followed by the addition and subtraction.

$$8+6 * 2/3-(-3) \qquad (3\text{-}1)$$

The same would be written in Racket as shown in code example (3-2).

$$(-\ (+\ 8\ (\ *\ \ 6\ (/\ 2\ 3)))\ (-\ 3)) \qquad (3\text{-}2)$$

Racket treats mathematical operators as functions. Each function must be written in prefix notation and enclosed in parentheses. Where all expressions are written with parenthesis indicating the order of evaluation, it was not expected that students would improve in understanding order-of-operations as a result of participating in a Bootstrap course. The problem was written as shown in figure 3-3.

Solve the following expression (show your work): $5+1*4-8/2$

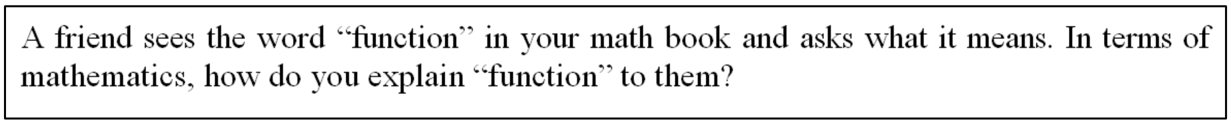
Figure 3-3 PEMDAS Question

The problem was scored on a range from 0 to 2 points. One point was given for the correct answer; another point was given point if the student demonstrated how the order-of-operations was applied.

3.2.3 Questions on Functions and Transfer

The assessments contained six questions concerning functions. Some of the questions on functions also queried the transfer of the advanced concepts of function composition and piecewise functions. Those questions were scored under both the functions and transfer categories. On the VHMS study, the two last questions were scored as transfer. After some evaluation, at the subsequent LHS study four of the six questions were scored as part of the transfer category. All the following questions were designed to assess how well students understood the concepts around functions that could be learned from the Bootstrap course.

The first question on functions, shown in figure 3-4, challenges the student to give a description of a function.



A friend sees the word “function” in your math book and asks what it means. In terms of mathematics, how do you explain “function” to them?

Figure 3-4 Function Definition Question

The question was scored on a scale of 0 to 4. According to the rubric, reference to input and output would give one point. A correct elaboration of a “machine” would give two additional points. To receive the 4th point, the student must give an exceptional answer. The following are examples of answers that received full credit: “A function is something used to find how a number relates to others,” “A function is something that two or more pieces of data are affected by. Function is labeling a + sign in the example 2+2=4,” and “What you put in a number you get a new number.”

The next question tested a student’s understanding of functions, variables, and composition. The question reflects back to the first question on variables, where the student was asked if a variable could represent an expression using a different variable. Here the concept is taken an additional step, where the student is expected to use the concept of a variable representing an expression in the context of a function. Because the Bootstrap curriculum requires students to nest expressions to build a function, the question was also scored as a transfer question at LHS.

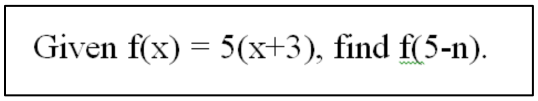

Given f(x) = 5(x+3), find f(5-n).

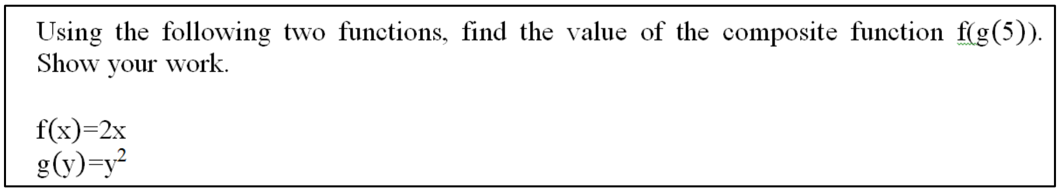
Figure 3-5 Functions/Variable Replacement Question

Equation (3-3) shows the expected answer.

$$f(5\text{-}n) = 5((5\text{-}n)+3) \tag{3-3}$$

If the student used function composition, then one point was awarded. Two more points were given if the student answered the question correctly.

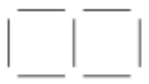
The problem shown in figure 3-6 asked the student to perform function composition with two functions. The students performed several similar operations in programming in the Bootstrap courses, where functions were built from other functions. Students could receive up to 3 points for their answer, one point for using g(5), and two more points for the correct answer. This question was included in the transfer score at the LHS study.


Using the following two functions, find the value of the composite function f(g(5)). Show your work.

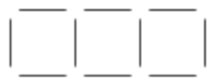
$f(x)=2x$
$g(y)=y^2$

Figure 3-6 Composition of Functions Question

If a side of each square in the shape below is 1 toothpick, then it takes 7 toothpicks to make 2 squares in a row.



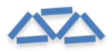
A. How many toothpicks are required to make 3 squares in a row?



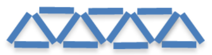
B. How many toothpicks are required to make 7 squares in a row? Show your work.
C. Describe in words how you could figure out how many toothpicks are needed to build any number of squares in a row.
D. Joan built 30 squares in a row. How many toothpicks did she use? Show your work.
E. Tom built a row of squares using 151 toothpicks. How many squares did he build?

Figure 3-7 Pre Assessment Toothpick Question

If a side of each triangle in the shape below is 1 toothpick, then it takes 7 toothpicks to make 3 triangles in a row.



A. How many toothpicks are required to make 7 triangles in a row?



B. How many toothpicks are required to make 20 triangles in a row? Show your work.
C. Write a function to demonstrate how you could figure out how many toothpicks are needed to build any number of triangles in a row.
D. Joan built 50 triangles in a row. How many toothpicks did she use? Show your work.
E. Tom built a row of triangles using 457 toothpicks. How many triangles did he build?

Figure 3-8 Post Assessment Toothpick Question

The problems shown in figures 3-10 and 3-11 were the last questions on the assessment. The purpose of the question is to test the students' ability to describe a situational problem as a function and use that function to find other answers. Part A gives context to the whole problem. Because it was expected the student would count the toothpicks to answer the question, part A was not scored. Students may try to answer part B through drawing the shapes and counting the toothpicks. That act was intended to help the students answer part C. Once a student sees the pattern to create the shapes, the student may be able to piece together a function to answer the rest of the questions. Parts D and E are designed to be sufficiently large such that a student could not answer the question without creating the function in part C. Parts B and C were scored on a 5 point scale each. Parts D and E were scored on a 2 point scale each. Only part C was categorized under functions. The rest of the problem was categorized under other algebra questions.

The following question tested the students' ability to use the Pythagorean Theorem and Cartesian Coordinates. Three points were possible for part A and part B. On part A, one point was given for drawing the x and y axis, one point for labeling the point f(0) on the graph, and one point for labeling the point f(4) on the graph. For part B, the student was awarded for using either the Pythagorean Theorem or the distance formula, and an additional point for achieving the correct answer. Part A tested students' ability to read a function and graph two points. It was expected that the students learn about Cartesian Coordinates from working with a graphical environment. Near the end of the course, students were required to use the Pythagorean Theorem to calculate the distance between two points on the screen. To answer part B, the student needed to recognize the algebra syntax to define a function, f(x), and use the function to calculate the value for two inputs.

Consider the following function:

$f(x)={}^{3}/_{4}x+2$

A. On a graph, plot and label f(0) and f(4).

B. How far apart are the two points? Show your work.

Figure 3-9 Pythagorean Theorem Question

Because in the VHMS course most students did not attempt to answer this question, the question for the high school course was changed as shown in figure 3-10.

Below is a function written in the Racket programming language. The name of the function is *f*, and it takes one parameter called *x*.

```
( define(f x)  (-(* (÷4 3) x)1))
```

The same function can be written in algebra notation.

$$f(x)={}^{4}/_{3}x-1$$

A. On a graph, plot and label f(0) and f(3).
B. Using the Pythagorean Theorem ($a^2+b^2=c^2$), find how far apart are the two points? Show your work.

Figure 3-10 Updated Pythagorean Theorem Question

In the change, the students are presented with the function written in both Racket and algebra syntax. The purpose of the change was to prompt the students to make the transfer from their work in Bootstrap to algebra.

The following question on function and transfer of concepts deals with a piecewise function.

Graph the following piecewise function

$$f(x)=\begin{cases} {}^{3}/_{2}x+2 \text{ for } x<2 \\ -{}^{1}/_{2}x+2 \text{ for } x\geq 2 \end{cases}$$

Figure 3-11 Piecewise Function Question

The students were taught conditional functions in Racket, which parallel piecewise function in algebra. This question was scored on a 3-point scale. One point was given for drawing a graph, another point for showing $x<2$ on the graph, and another point for showing $x\geq2$ on the graph.

3.2.4 Other Algebra Questions

The first general algebra question tested students understanding of balanced equations with variables.

Consider the following number sentence: $n - 84 = 227$
The value of n that makes this number sentence true is 311, because $311 - 84 = 227$
If 10 is added to each side of the number sentence, the new number sentence is as follows:
$n - 84 + 10 = 227 + 10$
What value of n makes this new number sentence true?
Please explain how you got your answer.

Figure 3-12 General Algebra Question

The question was scored on a 2 point scale with partial credit as an option. One point was given for giving the correct answer for the value of n (which was 311). Another point was awarded for showing algebraic manipulation to isolate n.

The following questions are about the expression:

$$\begin{array}{c} 3+4\underset{\uparrow}{=}7 \end{array}$$

A. The arrow above points to the = symbol. What is the name of symbol?
B. What does the symbol mean?
C. Can the symbol mean anything else? If yes, please explain.

Figure 3-13 Equality Question

The question in figure 3-13 is from Knuth et al., (2005, p. 70). Part A asks the name so that the name will not give the answer to part B. The reasoning behind part C was to draw from previous experience other uses of the same symbol. Knuth et al. explain that students often give additional interpretations. The question was left out of the assessments for VHMS because equivalency was not covered in the bootstrap curriculum, except when paired with greater than or less than. At LHS, the question was added by the research team to expand the domain of the study.

3.3 Statistical Methods

The results from the assessments were analyzed using multiple linear regression, a type of multiple regression. Ramsey and Schafer (2002) explain that "multiple regression analysis is one of the most widely used statistical tools … it is remarkably effective for answering questions involving many variables" (p. 235). The regression is defined as a rule or equation, "that describes the mean of the distribution of a single response variable (Y) for a particular set of explanatory variables ($X_{1,}X_2, \ldots$)" (p. 240). There can be multiple models that describe the same regression, and the simpler the model the more useful it is. The model can be simplified by removing explanatory variables by determining if they do not significantly alter the results of the analysis. A model reduced to only the treatment variable and a binary categorical variable may be described as follows,

$$\mu\{Y \mid X_1, X_2\} = \beta_0 + \beta_1 X_1 + \beta_2 X_2 \qquad (3\text{-}4)$$

which is read as "the regression of Y on $X_1$and X_2." In this case, the regression can be represented on a graph by two parallel lines; the distance between the two lines is defined by β_0.

If X_2 is the categorical variable, then the regression calculates the probability that β_2-β_0=0.

When additional explanatory variables are found to significantly affect the regression, then because of the complexity of the model it is not presented graphically.

For this study, the JMP software by SAS was used in calculating the regression as well as providing tools to reduce the model. After using a stepwise variable selection technique, the Bayes Information Criteria (BIC) was used to determine an appropriate reduced model. An Extra Sum of the Squares F test was then used to verify that significant variables were not eliminated.

In this study, the regression calculated the probability of the difference of the post assessment scores between the treatment and control groups based on the pre assessment scores of both groups. The pre assessment score and membership in the treatment vs. control group were included as explanatory variables. Without any other explanatory variables, the model could be described as follows,

$$\mu\{\text{post assessment scores} \mid \text{pre assessment scores, treatment vs. control group}\} \qquad (3\text{-}5)$$

which is read as the "mean post assessment scores as a function of pre assessment scores and treatment vs. control group."

The demographic information was also included as the explanatory variables, although were analyzed to determine significance with respect to the regression. The demographic information that was collected included age, current grade in school, gender, and current math class. Additionally, the responses to three questions were included as explanatory variables. The students were asked to rank themselves on a scale of 1 to 7 on the following questions: (1) "How do you feel about your programming skills?" (2) "Rate how quickly you learn new technologies?" and (3) "How do you feel about your math skills?" For most of the regression tests in this study, all but the two required explanatory variables were determined to not significantly contribute to the regression model and were removed.

To better convey the results to the educational research community, an SMD effect size was also calculated. The SMD effect size measures the magnitude of the differences between mean scores of the treatment and control group. It is calculated by dividing the difference of the mean scores of the treatment and control group by the pooled standard deviation. Cohen's (1992) method of categorizing effect sizes was applied. Cohen categorized effect sizes as small, medium and large; small being 0.20, medium being 0.50, and large being 0.80. These categorizations give a general estimate as to the value of the statistic.

3.4 Interviews

The interviews in this study were designed to give additional insight beyond assessments. The interviews were given towards the end of the Bootstrap course, so that all the material covered in the interview had been presented to the participant. The interviews were based on a demonstration by Schanzer, where he coached a student through relating function composition in Racket to algebra. The intent of the interviews was to observe if and how the student transferred the concept from Racket to algebra, without coaching. The questions were designed to demonstrate the participants' understanding of the definitions of variables and functions, and how the concepts transferred between the two disciplines. The interviews followed a predefined script (see appendix C).

During the interview, Racket coding samples and mathematical expressions were written on a whiteboard. The interview was audio recorded with a recording pen. Some prompting occurred if the student did not understand the question, or it was assumed the student might make a cognitive connection with a little prompting. There was no pressure put on the participant during the interview. If a participant did not know the answer, the interviewer indicated that not knowing is an acceptable answer, and moved on with the next question. If it was clear that the student would not be able to understand the rest of the questions, then the interview ended at that point. To end the interview on a positive note, the interviewer explains to the interviewee that the questions come from advanced mathematical concepts that they are not expected to know at their current educational levels.

The interview questions were divided into two sets. The first set examines the participants' understand of the relationship between functions and variables, both in Racket and in algebra. The second set looks at the participants' ability to transfer the composition of functions from Racket to algebra.

In the first set of questions, the participant was shown a function definition in Racket, and several functions using the newly defined function. A similar function in algebraic notation was presented. The question was meant to determine if the participant understood that the only difference between the function in Racket and algebra is syntax.

The second set of questions delved deeper into the understanding of functions, and the relationship between functions and parameters. The Racket functions showing composition were meant to help the participant if he or she did not understand the example of algebraic composite functions. The interview ended with asking the interviewee to define a variable, a function, and the relationship between the two.

The interviews at DMS and VHMS were coded and categorized according to Merriam (1998) and Charmaz (2002) (see appendix E). The responses were coded with action codes, and categories were created from the common themes that emerged from the action codes. The responses were then placed into a table under the category to which the response belonged. The action coding was listed as the properties of the category. Hypotheses about the responses were created from the properties and categories. The properties and hypotheses of the interviews were then searched for similarities. Since the interviews were divided into the same categories, each category was compared across the interviews to find similar themes. The general location of the themes was compared within the structured interview itself. From identifying the themes and their location, common patterns were deduced from the interviews.

Further interviews were conducted with four students LHS. The interviews were adjusted from the original script based off the experience gained from the previous analysis. The goal of the interviews at LHS was to observe if each student could make the connection between function composition in Racket and function composition in algebra, the same with piecewise functions. The amount of coaching was not restricted. The only inhibitors were time and the students' ability to make the connection. Because the interviews were taking time away from the class, these interviews were limited from 5 to 10 minutes each.

3.4.1 Participants

Two students from Dixon Middle School were interviewed. The first student, named Pedro (names of youth are changed), was 14 years old, and in eighth grade. At the time of the interview, he was completing his first year of Algebra. He was born in Latin America, and now is a U.S. citizen. He stated that English is his primary language. At the beginning of the interview, Pedro stumbled over understanding a function written in Racket. To help, the interviewer typed the function into the computer and ran it. Pedro subsequently was able to identify various outputs from given inputs. He readily caught the notion of how a function written in Bootstrap and a function written in algebraic notation could represent the same thing. Further during the interview, he could follow writing algebraic functions in Bootstrap, but never understood composite functions. He defined a variable as “a number”, but could not verbally define a function.

The second participant, Tyler, was 13 years old and was completing the seventh grade. He was finishing the first half of Pre-Algebra. Tyler quickly recognized the operation of the initial function written in Racket. He had not seen algebraic functional notation before, but was able to understand it with some coaching. After teaching him how the same function can be written algebraically and in Bootstrap, he seemed to understand the concept. But like Pedro, Tyler did not grasp the concept of composite functions.

The next student, named David, was interviewed during the last class of the VHMS Bootstrap course. He was selected because he finished his work early; and had time at the end of class for an interview. David was a bright student in algebra, and was a math tutor in his class. He was 13 years old, and was completing the seventh grade. David readily understood the Racket function written on the board and how it works. During the interview, he realized that Racket functions and algebraic functions could be the same thing, just different notation. Like the other interviews, David did not understand composite functions in algebra.

The first student at LMS, Sally, had completed Algebra I and said her current math class was personal finance. She was 17 years old, and in the 12th grade. Sally readily understood the definition of functions in Racket, and could readily decipher the code to state the result. But each time the discussion was moved to transfer to algebra, she declared she was confused. She indicated that function composition could be the same thing in both Racket and algebra, but she was very uncomfortable with algebraic syntax. She was able to discuss conditionals, a topic the previous three interviewees did not grasp. But when shown a piecewise function, she explained that it was very confusing. The question was asked, “If you were back at a math class, and you started working with this, would you feel more comfortable now?” She indicated, “No. Like I could like do it, but I think it would be hard because I learned like that. But I want to do it a different way.” It appeared that Sally had become comfortable with the syntax and concepts, but either disliked and/or feared the algebra notation.

The next student interviewed was named Ned, who was also 17 and in the 12th grade. He was concurrently taking Algebra I. Though Ned struggled with some basic arithmetic issues, he was able to work with both composition and talk through conditional functions written in Racket, something that had not happen in any of the interviews before. But he was not able to work with algebraic notation.

Dan was 18 and had taken Algebra I a couple of years ago. Dan recognized functional notation, but was not able to be coached that Racket functions and algebra functions were different only in syntax. Instead, he would say, “they are the same thing, but different answers.” The interviewer questioned Dan twice on this issue, and he gave the same response both times. Dan was also able to talk through the conditional functions in Racket notation.

The last student interviewed was Mike, who was 15 and currently taking Algebra I. Mike struggled through talking about functions in Racket syntax. As the interview progressed, it became clear that Mike was still struggling with arithmetic, giving out random answers. He indicated that he was not familiar with algebraic functional notation, and backed away from talking about conditional functions.

3.5 Delimitations

Establishing inference and determining cause and effect were not possible in this study. The selection of students in the control and experimental groups were not random. For two studies, members of the experimental group were chosen by volunteer, i.e. those willing to participate in and complete the course. In one course, students were selected from a category of students who were struggling in math, and were able to attend the course at the given time slot in the class schedule. Members of the control group consisted of students in a class where the instructor was willing to administer both pre and post evaluations. From that set of students, only those who returned their parental consent forms, and wrote their names (or the same name) on both the pre and post evaluation were eligible for the study. Neither selection of the students nor assignment to treatment vs. control groups was random. While this study is therefore categorized only as observational, it can indicate a potential benefit.

There were also problems comparing the three different observed Bootstrap courses. All three courses occurred at different schools with different instructors and included different age groups. The first course included students in 6th to 8th grades. The second course included students in 7th to 8th grade. The third course was taught at a high school, and consisted of students in the 11th and 12th grades. Statistical tools were used to determine if the age, grade, math classes taken, gender, and self-evaluated mathematics capability are significant within the control and experimental groups.

4 RESULTS

The purpose of this study was to determine if participation in a course of Bootstrap increases students' understanding of algebra. The statistical analysis showed that students understanding of variables improved, with suggestive results that the students gained a better understanding of functions. The Bootstrap students' ability of transfer between Bootstrap and algebra were not statistically significant. In subsequent interviews, students were able to demonstrate understanding the relationship between functions written in Racket and algebraic notation, but generally were not able to relate the concepts of composition and piecewise functions between the two disciplines.

The implementation of the three courses was very similar. There were two instructors for each course, and the instructors closely followed the curriculum. A primary difference was the background of the students at LHS. The Bootstrap course at LHS consisted of students who were struggling in math, whereas the Bootstrap students at DMS and VHMS volunteered to participate and were generally proficient in their current math courses. The course at LHS was therefore adjusted to a longer duration to accommodate the Bootstrap students' needs.

4.1 Quantitative Results

For each category for each Bootstrap course, two charts may be shown. The first chart is the box-plot showing the difference in students' scores. As explained in Chapter 3, when only the pre-assessment score was shown to be significant, the linear regression chart for both the treatment and control groups are shown.

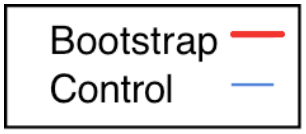


Figure 4-1 Legend for Regression Charts

4.1.1 Results for Vista Heights Middle School

Table 4-1 Statistical Analysis for Bootstrap Course at Vista Heights Middle School

Category	***Multiple Regression p-value***	***Effect Size***
Variables	.0009	.91
Functions	<.0001	.54
Transfer	.21	.43
PEMDAS	.94	-.12
Other	.02	.02

At Vista Heights, 9 students completed the Bootstrap course and 17 students were in the control group, giving N=26 for the multiple regression and effect size calculations. Table 4-1 shows the multiple regression p-values and the effect size for each category.

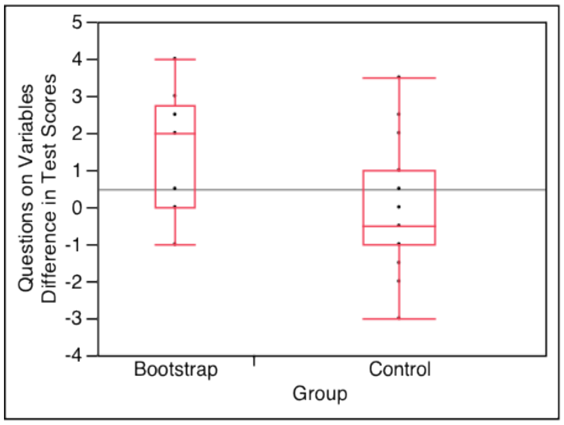


Figure 4-2 VHMS Box Plot for Variables

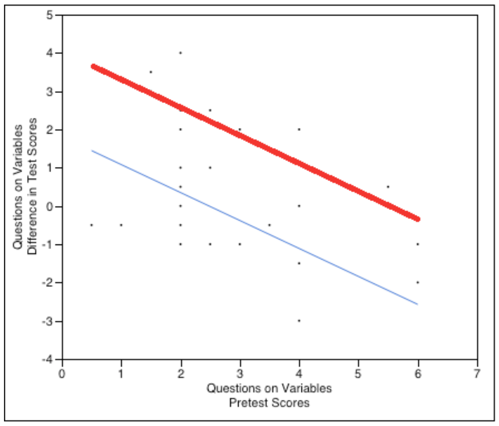


Figure 4-3 VHMS Regression for Variables

The results shown in table 4-1 indicate the Bootstrap students gained statistically significant improvement in the understanding of the categories of variables.

Figures 4-2 and 4-3 show the box plot and regression analysis for the variable category. The regression in figure 4-3 shows a difference of two points between the experimental and control groups, for which a p-value of .0009 was calculated; shown in table 4-1. The effect size results concurs findings concur with the multiple regression finding, demonstrating a Cohen's "large" effect size of .91.

The Bootstrap students also demonstrated a significant increase in understanding in the functions category, as shown in figures 4-4 and 4-5. On figure 4-4, the control group scores show some concern, where the box-plot for the control group is skewed down, and all but three students regressed in their scores. This skewing may be the reason for the extremely low p-value of <.0001. Figure 4-5 shows a difference of 5 points between the two regression lines. From table 4-2, the medium effect size of .54 gives a safer estimate of the general improvement of the Bootstrap students.

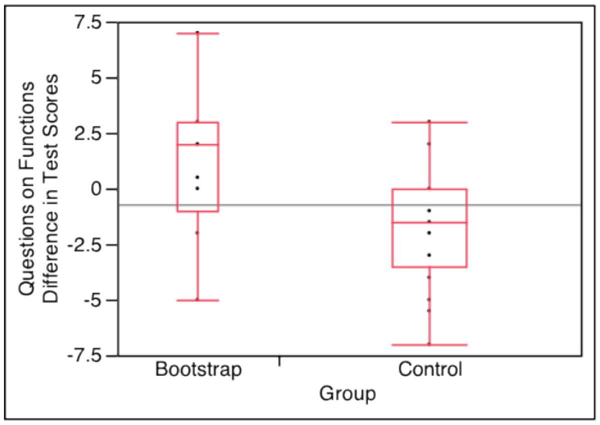


Figure 4-4 VHMS Box Plot for Functions

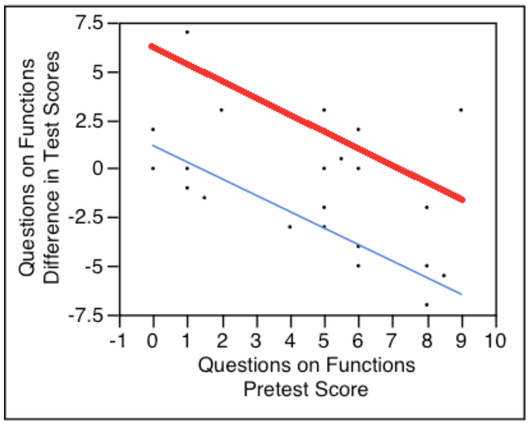


Figure 4-5 VHMS Regression for Functions

For the category consisting of other algebra problems, Figure 4-6 shows a modest improvement for the control group scores. Meanwhile the Bootstrap group scores were highly diverse.

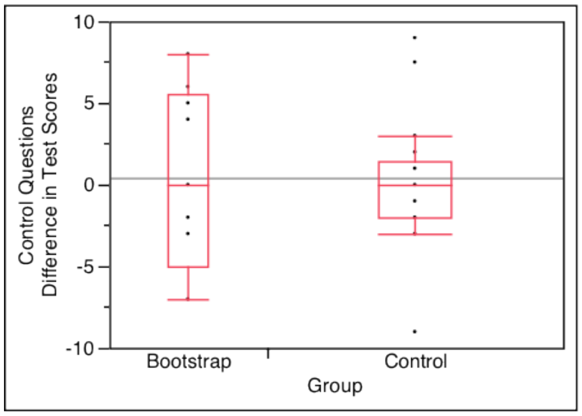


Figure 4-6 VHMS Box Plot for Other

The regression results shown Table 4-1 are suggestive but inconclusive, where the p-value of .02 falls outside of the Bonferroni adjusted significant p-value of 0.01. Also from table 4-1, the small effect size of .02 shows an answer that would be expected from the box plot.

For category of other algebra problems, the age, gender, current math class, self-evaluated programming skills, and the self-evaluated ability to learn new technologies were all shown to be significant explanatory variables in the regression. Because of the presence of these explanatory variables, the regression chart was not generated.

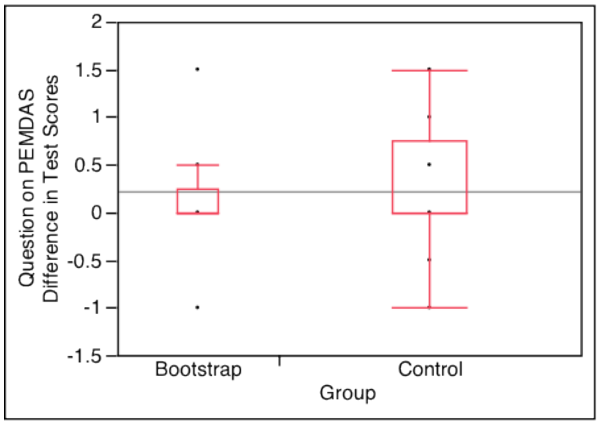


Figure 4-7 VHMS Box Plot for PEMDAS

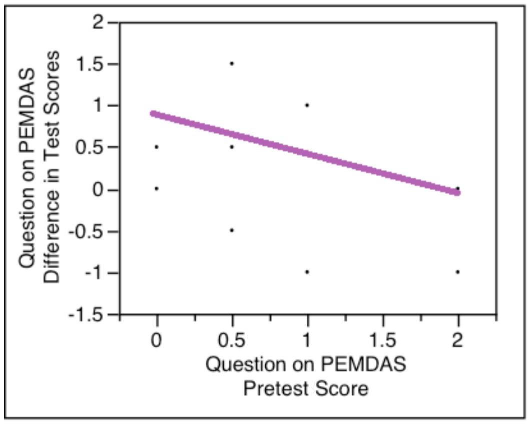


Figure 4-8 VHMS Regression for PEMDAS

As explained in chapter 3, it was not expected that the student would improve in the category of order-of-operations. The analysis shown in figures 4-7 and 4-8 shows the experimental group performed about the same as the control group. From table 4-1, the effect size of -.12 also demonstrates no general improvement for the Bootstrap students. The outlier that showed remarkable improvement from the Bootstrap group, shown in figure 4-7 and figure 4-9, was an advanced math student.

The statistics show that the students in the Bootstrap course did not significantly improve in the category of transfer from Bootstrap to algebra over the results from the control group. The medium effect size of .43 shown in table 4-1 should be accepted with caution. All but three student answered the questions on the post assessment, two from the Bootstrap course and one from the control group; as can be seen in figure 4-9.

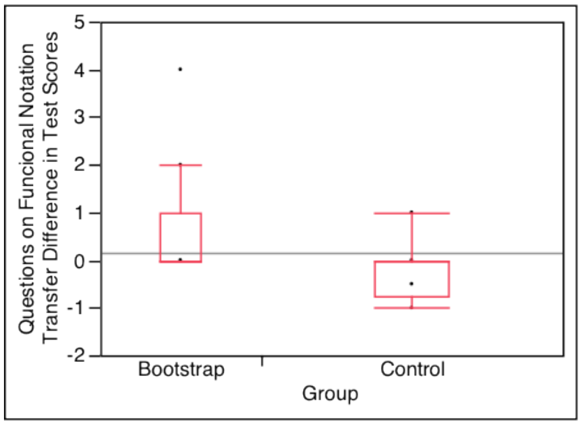


Figure 4-9 VHMS Box Plot for Transfer

On the transfer category, the current math class showed to be significant. This significance comes from the advanced math student, whose score is the outlier on figure 4-9.

4.1.2 Results for Lehi High School

Tables 4-2 shows the regression and effect size results for the LHS study. The effect size results are similar to the results in the VHMS study, but the multiple regression shows a less significant difference between the treatment and control groups in the categories of functions and transfer.

Table 4-2 Statistical Analysis for Bootstrap Course at Lehi High School

Category	*Multiple Regression p-value*	*Effect Size*
Variables	.0024	.93
Functions	.09	.52
Transfer	.15	.88
PEMDAS	1	1.26
Other	.0017	1.24

Data from table 4-2 and figures 4-10 and 4-11 shows that students at LHS Bootstrap course demonstrated a significant improvement in the understanding of variables.

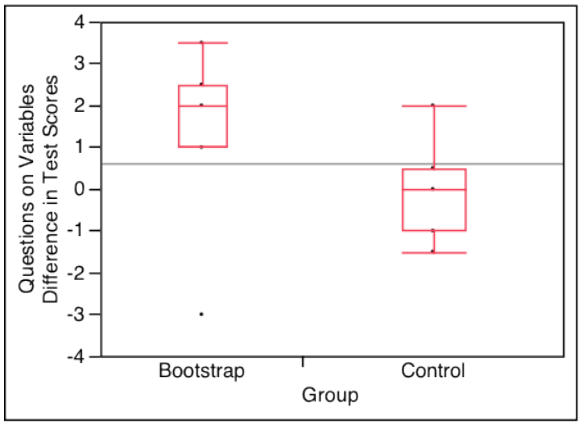


Figure 4-10 LHS Box Plot for Variables

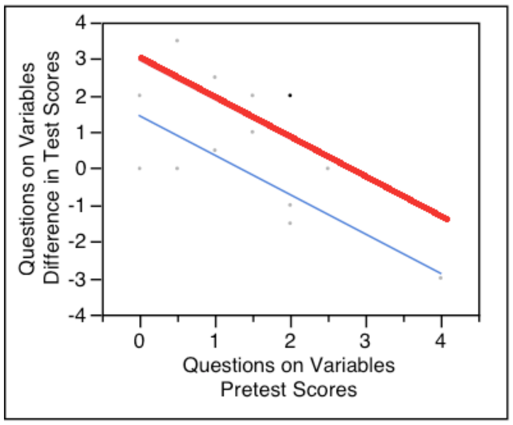


Figure 4-11 LHS Regression for Variables

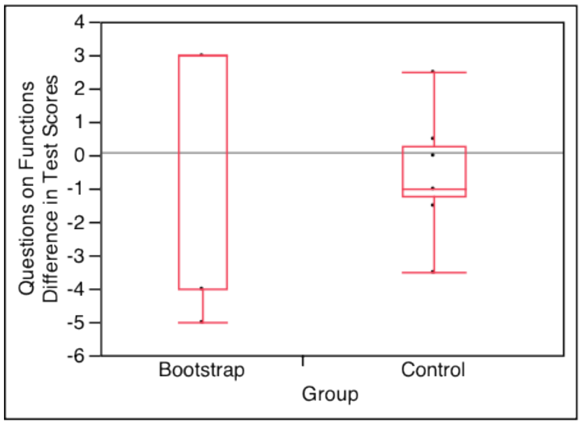


Figure 4-12 LHS Box Plot for Functions

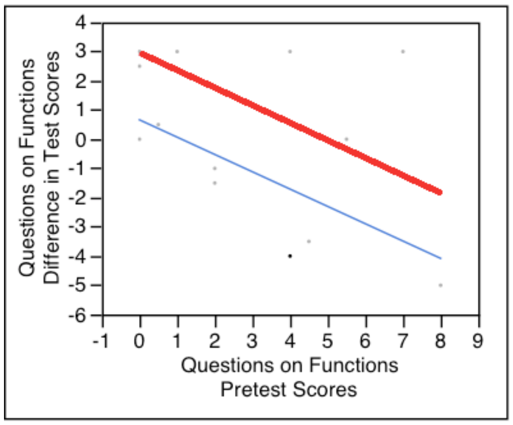


Figure 4-13 LHS Regression for Functions

Figure 4-12 shows a box plot for the Bootstrap group with the median at the top, indicating most did well, with a couple doing very poorly. The regression in figure 4-13 shows a clear difference between the groups, and the medium effect size of .52, found in table 4-4, is also indicative of significant improvement.

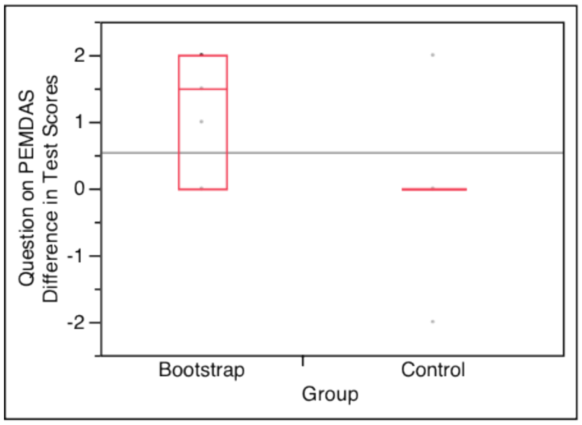


Figure 4-14 LHS Box Plot for PEMDAS

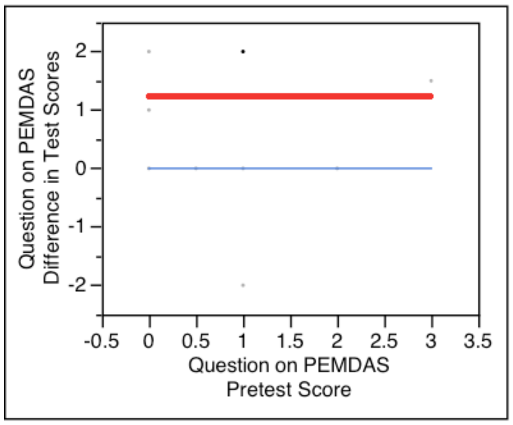


Figure 4-15 LHS Regression for PEMDAS

The statistics for the order-of-operations category are misleading. Table 4-2 shows a p-value of 1 and t a large effect size of 1.26.

Figures 4-14 and 4-15 show the reason behind the conflicting results. Figure 4-14 shows the control group to be flat, the two outliers cancelling each other out. Given that the average for the control group was 0, the situation of a p-value of 1 (as shown in table 4-2) is created, as shown in figure 4-15.

On the category of other algebra problems, figures 4-15 and 4-16 show a strong improvement on the part of the Bootstrap students. The statistics concur, where Table 4-2 indicates a p-value of .0017, and table 4-4 shows a large effect size of 1.24.

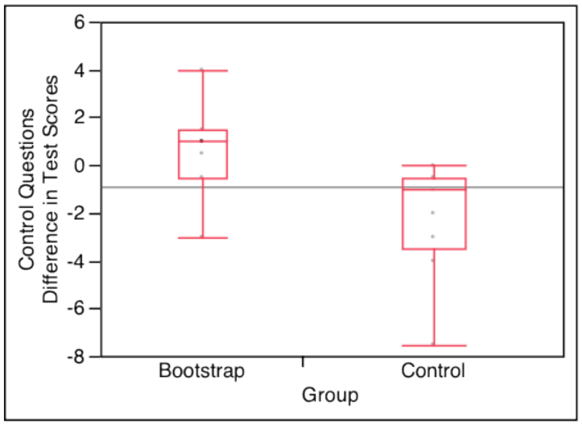


Figure 4-16 LHS Box Plot for Other

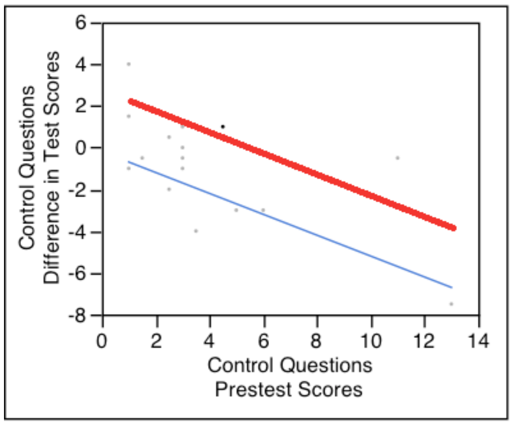


Figure 4-17 LHS Regression for Other

For the category of transfer from programming to algebra, Figure 4-18 shows reasonable improvement of up to 3 points for the Bootstrap students. The improvement was from question 8 (see appendix A.4), which the students left blank on the pre assessment but answered on the post assessment. Table 4-2 shows a weak p-value of .15, which is indicative but not conclusive, whereas the effect size from table 4-2 shows a strong .88. The mixed results should be considered cautiously.

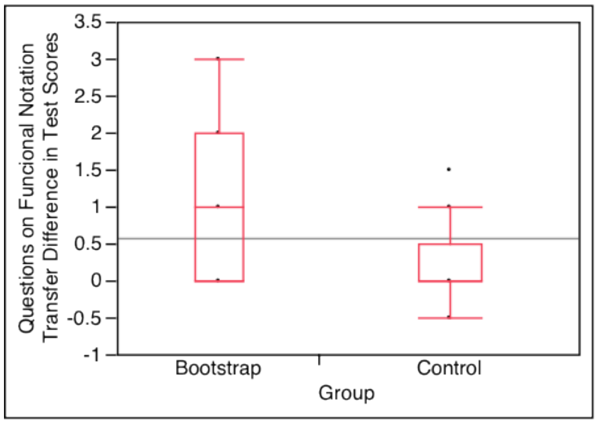


Figure 4-18 LHS Box Plot for Transfer

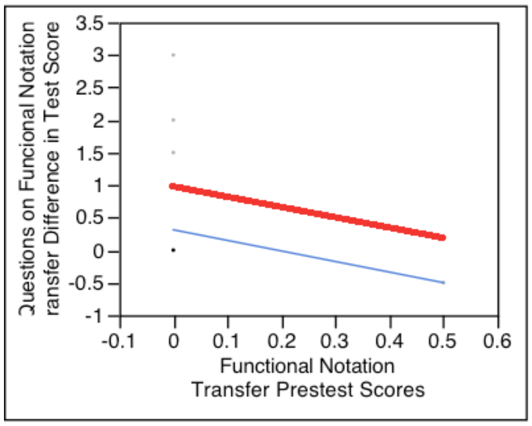


Figure 4-19 LHS Regression for Transfer

4.2 Qualitative Results

Three categories emerged from the coding of the interview responses: current math level, cognitive-bridge, and lack of understanding. Participants struggled with the concepts being presented throughout the interview. Most of the confusion was resolved with simplification and expansion of the questions. The participants were able to relate the algebraic concepts from Bootstrap based on their current understanding of mathematics. From there, the students made some connections between programming and algebra. But while the participants appeared to be comfortable with the understanding of functions, the participants were reluctant to put forth the effort to understand more complex relationships. The subsequent interviews from LHS generally followed the same patterns found in the interviews from DMS and VHMS.

In general, the participants demonstrated that the work they had done in the Bootstrap classes had prepared them for recognizing the operation of simple functions written in Racket language. Most of the students of Bootstrap that were interviewed were able to readily pick up working with simple functions written in Racket. Exceptions were Pedro, who required prompting using the familiar environment of the computer, and Ned, who needed some explanation regarding the structure of the Racket function. Mike never appeared to have understood the Racket functions, and needed full promptings of each function presented.

The theme of using previous mathematics experience proved consistent throughout the interviews. From the coded interviews, only Pedro had seen algebraic functional notation before. Where David and Tyler needed coaching through the section of the interview covering the comparison of the algebraic and Racket syntaxes, Pedro quickly recognized the similarity. It was clear that Pedro had not thought of the correlation before the interview, but understood it very rapidly. Where all three participants did grasp the concept, it appeared that both their algebra and Bootstrap experience prepared them for making the connection. The same was demonstrated with the interviews at LHS. Sally and Dan had already completed Algebra I, and were able to quickly make the connection between functions written in Racket and algebraic notation. But Mike and Ned, who were concurrently studying Algebra I with the programming course, had difficulty or were unable to grasp the correlation between the Bootstrap and algebra functional notation.

The cognitive bridge theme was demonstrating when each participant was able to relate functions in Racket to functions in algebra. Though Pedro struggled earlier in the interview, he was able to convert times3(x)=3x to f(x)=3x. The cognitive bridge came when he then realized that f(x)=3x was the same as (define f x (∗ 3 x). The interview with Tyler took a different track from the others in that Tyler did not recognize the correlation until the discussion of composition of functions. But at that time, he was able to write (+ (f(x) 5)) in response to moving from algebra composition of functions to notation in Racket. With some prompting from that point, he showed understanding of the equivalence of functionality outside of syntax. From going through the steps outlined in the interview script, David came to the following conclusion,

> One is a computer function, and one is a notation in math. In this case, it's saying that whatever you put in for value will be multiplied by 3. It's pretty much the same thing in the other case, it's just in a math format. (Appendix C.2.1.1)

Sally showed the process of understanding as she thought out loud. When asked what the difference between the function in Racket and algebra notation she said, "I… don't know… so like the x is over there, and it's in its own parentheses" (Appendix C.3.1). Sally stated that she had seen the algebra notation in a math class, but had only seen the Racket syntax in the Bootstrap course. She then was able to declare that both were the same thing.

The participants were generally able to discuss the topics of composite and piecewise functions in terms of the Racket programming language, but showed confusion and hesitancy when it came to the algebraic syntax for those two concepts. Throughout the interview, students were far more comfortable with the Racket syntax than algebraic notation. One reason may be due to having recently worked with Racket. Another reason shown in the interviews is that many students had only been recently introduced to algebraic notation. Tyler, Ned, and Mike indicated that they were not familiar with a function definition in algebraic notation. Sally declared that though she was familiar with the notation, she described it as scary. She indicated that she preferred the Racket syntax because of her recent work and success with the language. Pedro recognized an algebra definition of a function, but quickly was lost on composition. David showed similar comprehension. None of the participants were able to answer questions with the algebraic notation of piecewise functions. But Sally, Ned, and Dan were able to converse about conditionals in Racket.

4.3 Discussion

The statistical results showed a strong improvement in the understanding of variables at both VHMS and LHS Bootstrap courses. The improvement in the understanding of functions at both schools was also noteworthy. The effect size results for the Bootstrap participants at both schools were large in the category of variables and medium in functions. The interviews reflected these results where students of Bootstrap were able to converse about functions and variables, and relate them to algebraic syntax. The statistical results at both schools showed that neither treatment group gained a significantly greater understanding of order-of-operations. This result was anticipated in that the concept of order-of-operations was outside the scope of the Bootstrap curriculum.

But another pattern that emerged from both the assessments and interviews was the difficultly the Bootstrap students had with the composition of functions and piecewise functions in algebraic notation. At both schools, the statistical results for the transfer category for the Bootstrap participants were mixed. In the regression analysis, both showed no significant improvement on the part of the treatment group over the control group. But VHMS Bootstrap students showed medium effect size, and the LHS Bootstrap students showed a large effect size. A concern of the assessments were that most students, both treatment and control groups, left most of the transfer questions blank; leaving in doubt the validity of the effect size results. On both the pre and post assessments, none of the treatment group at both schools answered the graphing function that required use of the Pythagorean Theorem. Only two students (one from the treatment group and the other from the control group) from LHS answered the piecewise function question, but only on the post-assessment. All other students in both control and treatment groups, left the question blank. Only one student answered the question shown in figure 3-5 on the post assessment for both schools. No students answered the question on the pre-assessment.

The participants who were interviewed gave various reasons as to why they did not attempt to answer these questions. The common theme from the interviews on these questions dealt with unfamiliarity with algebraic notation. The instructors for the LHS course specifically discussed composition of functions and conditionals and how they relate to algebra. But given the difficulties the students showed with the concepts in both the assessments and interviews, it seems that more than one lecture during a class is required for the students to grasp the concepts. Sally stated concerning the unanswered questions, "I had no idea what I was doing on that part" (Appendix C). In composition of functions, the nesting of the functions on the left hand side of the equal sign was also confusing to David and Sally. David interpreted $f(g(x))$ as multiplying f and g. None of the interview participants were familiar with the notation for piecewise functions. Concerning a piecewise function Sally said, "It just looks like it is written in a different format, a lot different than you write in" (Appendix C). The piecewise function question, and a question that required using the Pythagorean Theorem, both required the student to graph a function on the grid. Both Tyler and Sally indicated that they did not understand how to graph a function.

5 CONCLUSIONS AND RECOMMENDATIONS

The intent of this study was to statistically measure the effect of participation in a course of Bootstrap on a student's understanding of the algebraic concepts of variables and functions. Qualitative interviews were also conducted to give additional insight into the pathways and obstructions students encountered when trying to transfer the experience of functional programming to algebra. The data suggests these students did gain a significantly greater understanding of functions and variables over the control group. In interviews students were generally able to relate functions in programming to functions in algebra, but were not able to transfer their experience in the advanced topics of function composition and piecewise functions from programming to algebra by the end of the course. Other emergent themes presented themselves during the duration of the studies. A combination of these themes as well as the results may be useful in presenting the curriculum to other schools for implementation. Further studies may consider focusing on testing algebraic concepts rather than syntax. Studying the longevity of newly acquired skills would also be helpful.

5.1 Summary of Findings

The statistical data showed that the students understanding of variables improved over the time of the course. The data from VHMS showed improvement on the understanding of functions, while the data from LHS was suggestive but inconclusive. Transfer of understanding between the two disciplines was clear in terms of understanding of functions and variables. Further, the interviewed participants showed a nearly consistent ability to relate functions in Racket to functions in algebra. The students of Bootstrap had difficulty in relating the concepts of function composition and piecewise functions from programming to algebra. Most participants did not answer those questions concerning those topics on the assessment. When asked about it in interviews, the response generally was that the notation was unfamiliar and confusing. Even when the students had been coached on algebraic notation beforehand, the students still were not able to connect the concepts on their own. It was clear that the students understood and readily used the concepts when programming, but cognitively the students seemed to have attached the concepts to the syntax of the Racket language. Further research examining as to why Bootstrap students were able to better understand the concept of variables and functions at the end of the course would be beneficial. Likewise, future studies could examine specifically how students were able to utilize advanced algebraic concepts in programming, and research methods to transfer that experience algebra.

The Bootstrap curriculum, using direct instruction, presented lessons with step-by-step instructions for the instructors. Observational data was only collected at the end of each course when assessments and interviews were given. Therefore it is not possible for the researcher to verify if the courses were presented in the prescribed manner. As such, the variation in lesson delivery may account for the variability in results. The results of the function transfer walk-through exercises suggest that students were indeed capable of making the connection between functions in algebra and functions in programming. This revelation warrants further research and suggests that the implementation of the curriculum may be as important as the concepts taught themselves.

An example of how explicit instruction could aid students' understanding lies in the one-to-one correlation between the concepts presented in Bootstrap and corresponding algebraic concepts. Bootstrap students better grasped the concepts of variables and functions in algebra, even though the curriculum does not explicitly related the concepts between the two disciplines. This study anticipated that Bootstrap students would also match the concepts of function composition and piecewise functions implicitly. Further research could examine how explicit instruction correlating composite and piecewise functions between Racket and algebra better aid students' transfer of the two concepts.

5.1.1 Scope of Inference

Since the students were not randomly selected, we cannot infer these results to the general population. Further, because students were not randomly assigned to the treatment and control groups, cause and effect also cannot be established. But what was found from these two observational studies was the repeated success in increased understanding of variables and functions. Further similar studies may contribute to a better understanding of the pedagogical relationship between functional programming and algebraic concepts.

5.2 Benefit of Functional Programming

Felleisen and Krishnamurthi (2009) made the following statement concerning aligning education of mathematics with programming,

> The goal of an alignment is to transfer skills from programming to mathematics and vice versa. While students quickly grasp small differences in syntax, they will mentally block if the *notion* of, say, "function" in programming significantly differs from the notion of "function" in algebra. (p. 39)

The goal of alignment was shown to be successful in the Bootstrap student interviews in that the Bootstrap students were able to understand the mapping of function notation from Racket to algebra. Through a discussion showing how a function in algebra behaves the same way in Racket, all interviewed Bootstrap students except Mike were able to connect the two notations as the same concept. But the difference in syntax for composite and piecewise functions was enough to be troublesome for the treatment group in both the assessments and the interviews.

However, the students interviewed from LHS were able to discuss the concepts of piecewise and composition of functions. These were students who had a history of having troubles in math, and were selected for the course by the school administration. But at the end of the nine-week course these students were not having troubles with programming and discussing algebraic concepts, albeit in the context of functional programming. The goal stated by Felleisen and Krishnamurthi were not fully accomplished their definition of transfer, but the language, with its functional style, proved not to be a mental block.

But by virtue of writing the assigned programs, the Bootstrap students were using the advanced algebra concepts. In essence, the students were using algebra in a project-oriented environment. Though the syntax is different between algebra and Racket, the concepts are the same. The parameters in Racket behave the same as variables in algebra. Functions share the same definition in both Bootstrap's programming language and algebra. This could be the reason why Bootstrap students did well on both the variables and the concept of functions on the post assessments, but did not do well on the transfer portion.

5.3 Length and Intensity of Treatment

Generally the various studies of transfer between mathematics and programming consisted of a wide range of instruction/exposure time, but the studies usually consisted of many weeks into months. The Paz and Leron (2009) study with Scheme consisted 3 weekly hours of study over the course of the school year, yielding 90 hours of instruction. North's (2005) observations similarly consisted of an entire school year. The Logo study by Milner (1973) consisted of three 5-week phases, with two 40-minute sessions per week; totaling 20 hours of study over 15 weeks. The Logo study by Howe et al. (1980) lasted 1½ year with one hour of Logo study per week. The study by Noss (1986) consisted of approximately 50 hours over 18 months. Sutherland's (1989) research consisted of three years of instruction with limited results. Palumbo (1990) references two studies by Palumbo and Reed using BASIC to teach problem solving skills. These two studies consisted of 72 hours and 90 hours of exposure. Palumbo also references a Pea and Kurland study that taught Logo for 30 minutes twice a week, for a total of 30 hours of exposure. The study did not find any significant differences between the control and treatment groups. Palumbo also references Salomon and Perkins, who question "whether 30 hours of treatment is long enough to produce any type of training and transfer" (p. 81). Palumbo further states, "Because students do not become experts in traditional programming-language courses, it is little wonder that no low-road transfer of training exists" (p. 80).

The courses of Bootstrap at DMS and VHMS comprised approximately 20 hours of instruction over 6 weeks. The course at LHS consisted of approximately 40 hours of instruction over 8 weeks. Compared with similar studies, these Bootstrap courses consisted of a low amount of instruction time over a shorter duration.

5.4 Implications and Recommendations

This study has demonstrated aspects of the Bootstrap curriculum that may make it valuable to secondary education algebra instructors. Concerning the use of programming to teach mathematics, Johnson (2000) stated,

> While one can point to a number of exemplary and innovative classroom activities, is has often been the case that implementation has been fragmented with the focus being on learning the programming language for its own sake with little attention given to any longer term integration within the mathematics curriculum. The result of such a limited practice is that programming becomes just another new topic for teaching and learning to be 'squeezed' into an already 'full' curriculum with little or no apparent future 'pay-off'. (p. 202)

Johnson's argument is against focusing on the language and giving little attention to integration with mathematics. The Bootstrap curriculum suffers from neither of these conditions. Its express purpose is to teach algebra through programming (BootstrapWorld). The curriculum is designed to work as part of a mathematics curriculum. Schanzer states,

> Bootstrap uses algebra as the vehicle for creating images and animations. That means that concepts students encounter in Bootstrap behave the exact same way that they do in math class. This lets students experiment with algebraic concepts by writing functions. (Ibid.)

It was shown that there was strong benefit to the students of these Bootstrap courses in terms of understanding variables and functions. Further, the students were executing algebraic ideas, albeit in a different syntax than algebraic notation, and showed enthusiasm in doing so.

The weakness in this study on the lack of transfer to algebra notation may be due to the length and approach from this course. At both VHMS and LHS, students were instructed concerning function composition and piecewise functions, but the students of Bootstrap demonstrated a lack of understanding of these concepts using algebraic notation; both in the assessments and the interviews. It may be assumed that the duration for this study was too short for the intended goals for transfer from Bootstrap to algebra. It may have been not practical to expect beginning or struggling algebra students to be conversant in advanced algebra after a few weeks of programming. But the assessments were directed exactly to that goal. From the interviews, students demonstrated that the complex notation on the assessments deterred students from answering the questions.

Future studies should consider more practical goals based on the students' experience and the duration of the course. This study added the challenging questions in a hope to provide data to impress mathematics educators, and in so doing incorporate the curriculum into their schools. A better assessment would have fewer questions concerning algebraic syntax and more questions on understanding the nature of functions, particularly in the categories of composition and conditionals. If a study was based on an algebra course using Bootstrap as an introduction to functions, it may be valuable to have another assessment on algebraic syntax, as well as concepts, and compare those results to a control group consisting of an algebra course not using the Bootstrap curriculum.

In youth educational environments, it is difficult for research to show statistical inference and cause and effect. To demonstrate inference, the pool of students needs to be selected randomly from a population. To show cause and effect, the students need to be randomly distributed into the treatment and control groups. Studies such as this require participants willing to enroll in and complete a Bootstrap course. That very requirement eliminates the possibility of randomness in both selection from a population and distribution between the treatment vs. the control groups. Further reducing randomization, parents of the youth may want to move students in and out of the study classes to what the parents perceive as the class with the greatest benefit.

A possible study could occur at a middle school where an algebra instructor taught two beginning algebra classes, one using Bootstrap as part of the curriculum and another algebra class without Bootstrap. Assessments could be given to both classes at the beginning of the course, at the beginning of when the treatment class begins and ends the Bootstrap section, and at the end of the course. The study then could observe students beyond immediately after the end of Bootstrap instruction and continue to measure transfer to the end of the algebra course. A study so designed would eliminate the confounding variable of using different teachers for the treatment and control groups. Inference could only be applied to beginning algebra students at the school where the studied occurred. Random selection between the treatment and control groups is compromised in that students are able to transfer between classes. Further, the required parental consent may cause parents to transfer students in and out of the treatment class, compromising the demonstration of cause and effect. Nevertheless, a studied so designed reduces confounding variables and the number of explanatory variables. Several studies designed as explained could greatly aid mathematics educators in considering adopting the Bootstrap curriculum into their courses.

5.5 Summary Statement

The purpose of this study was to evaluate how participation in a course based on the Boostrap curriculum would affect secondary education students' understanding of algebra. The students' comprehension of variables, functions, and transfer from programming in Racket to algebra were analyzed using a multiple regression and effect size from scores on pre and post assessments. Observational studies were also performed as interviews with some students of the Bootstrap courses. The statistical analysis showed a significant improvement over a control group in scores on questions concerning variables and functions. The results concerning transfer between programming and mathematics were mixed, primarily because the students in both the treatment and control groups did not answer the questions on the assessments concerning piecewise functions and function composition. Interviews subsequently demonstrated student understanding of the concepts behind these advanced concepts, but showed a lack of transfer of that knowledge to algebraic notation. It was noted that the improvements the students of Bootstrap made occurred with less participation time than comparable studies. The use of the Bootstrap curriculum in this study has shown to give the students of Bootstrap of this study a project-based environment to learn algebra.

REFERENCES

Exploring Computational Thinking. Google. http://www.google.com/edu/computational-thinking/ (accessed 8/25, 2012).

Racket. http://racket-lang.org (accessed 11/10, 2012).

Bloch, Stephen, Clements, John, Felleisen, Matthias, Findler, Robby, Fisler, Kathi, Flatt, Matthew, Proulx, Viera and Krishnamurthi, Shriram. Teach Scheme -> Reach Java. http://teach-scheme.org/ (accessed 04/08, 2011).

Boyes, Geoffrey, Trevor Fletcher, David Johnson, Adrian Oldknow, Richard Phillips, John Warwick, Jo Watson, and William Wynne-Willson. 1985. Mathematics and microcomputers: A Pendly Manor Report. Mathematics in School 14, no. 2: pp. 26-28.

Charmaz, Kathy. 2002. Qualitative Interviewing and Grounded Theory Analysis. In *Handbook of Interview Research*, ed. J. Gubrium and J. A. Holstein. 1st ed., pp. 675-694 Thousand Oaks: Sage.

Clements, Douglas. 1985. Research on Logo in Education: Is the Turtle Slow but Steady, or Not Even in the Race? *Computers in the Schools* 2, no. 2-3: 55-71.

Clements, Douglas H. and Julie Sarama. 1997. Research on Logo. *Computers in the Schools* 14, no. 1-2: 9-46.

Cohen, Jacob. 1992. A Power Primer. (Power Analysis) (Quantitative Methods in Psychology). *Psychological Bulletin,* 155.

Felleisen, Matthias. 2011. TeachScheme! Dallas, TX, USA, ACM.

Felleisen, Matthias. 2010. TeachScheme!: A Checkpoint. *SIGPLAN Not.* 45, no. 9: 129-130.

Felleisen, Matthias, Robert Bruce Findler, Matthew Flatt, and Shriram Krishnamurthi. 2004. The TeachScheme! Project: Computing and Programming for Every Student. *Computer Science Education* 14, no. 1: 55-77.

Felleisen, Mattias, Robert Bruce Finder, Mattew Flatt, and Shriram Krishnamurthi. 2004. The Structure and Interpretation of the Computer Science Curriculum. *Journal of Functional Programming* 14, no. 04: 365.

Felleisen, Matthias and Shriram Krishnamurthi. 2009. Viewpoint: Why Computer Science Doesn't Matter. *Commun.ACM* 52, no. 7: 37-40.

Feurzeig, Wallace, Seymour Papert, Marjorie Bloom, Richard Grant, and Cynthia. Solomon. 1970. Programming-languages as a Conceptual Framework for Teaching Mathematics. *SIGCUE Bulletin* 4, no. 2: 13.

Feurzeig, Wallace. 1986. Algebra Slaves and Agents in a Logo-Based Mathematics Curriculum. *Instructional Science* 14, no. 3-4: 229-254.

Feurzeig, Wallace. The Logo Lineage. http://logoplus.pagesperso-orange.fr/private/The LOGO lineage.pdf (accessed 4/18, 2011).

Feurzeig, Wallace. 2010. Toward a Culture of Creativity: A Personal Perspective on Logo's Early Years and Ongoing Potential. *International Journal of Computers for Mathematical Learning* 15, no. 3: 257-265.

Findler, Robert Bruce, John Clements, Cormac Flanagan, Matthew Flatt, Shriram Krishnamurthi, Paul Steckler, and Matthias Felleisen. 2002. DrScheme: A Programming Environment for Scheme. *Journal of Functional Programming* 12, no. 02: 159-182.

Harvey, Brian. 1985. *Computer Science Logo Style*. 2nd ed. Vol. 1 MIT.

Hinsen, Konrad 2009. The Promises of Functional Programming. *Computing in Science & Engineering* 11, no. 4: 86-90.

Howe, Jim, A. M., Timothy O'Shea, and Fran Plane. 1980. Teaching Mathematics Through Logo Programming: An Evaluation Study. Roehampton, England, Amsterdam; New York: North-Holland Pub. Co.: sole distributors for U.S.A. and Canada, Elsevier North-Holland, 3-7 September 1979.

Howe, Jim A. M., Peter M. Ross, Ken R. Johnson, Fran Plane, and Ena Inglis. 1982. Teaching Mathematics Through Programming in the Classroom. Computers & Education 6, no. 1: 85-91.

Johnson, David C. 2000. Algorithmics and Programming in the School Mathematics Curriculum: Support is Waning - Is There Still a Case to be Made? *Education and Information Technologies* 5, no. 3: 201-201-214.

Kafai, Yasmin B. 2006. Playing and Making Games for Learning. *Games and Culture* 1, no. 1: 36-40.

Knuth, Eric J., Martha W. Alibali, Nicole M. McNeil, Aaron Weinberg, and Ana C. Stephens. 2005. Middle School Students' Understanding of Core Algebraic Concepts: Equivalence Variable. ZDM. Zentralblatt Für Didaktik Der Mathematik.Articles 37, no. 1: 68-76.

Krishnamurthi, Shriram, Matthias Felleisen, and Kathi Fisler. 1999. *The Technology that Computer Science Education Overlooked.*

Merriam, Sharan B. 1998. *Qualitative Research and Case Study Applications in Education*. Ed. Sharan B. Merriam. Rev. and expanded. ed.San Francisco, Calif.: Jossey-Bass.

Milner, Stuart. 1973. *The Effects of Computer Programming on Performance in Mathematics*.

North, Karen. 2005. Using DrScheme and the Design Recipe in Algebra. Brown University Providence, Rhode Island, June 11-12, 2005.

Noss, Richard. 1986. Constructing a Conceptual Framework for Elementary Algebra Through Logo Programming. *Educational Studies in Mathematics* 17, no. 4: 335-357.

Ortiz, Enrique and S. Kim MacGregor. 1995. Effects of Logo Programming on Understanding Variables. *Journal of Educational Computing Research* 7, no. 1: 37-1.

Palumbo, David B. 1990. Programming Language/Problem-Solving Research: A Review of Relevant Issues. *Review of Educational Research* 60, : 65-89.

Papert, Seymour. 2000. What's the Big Idea? Toward a Pedagogy of Idea Power. *IBM SYSTEMS JOURNAL* 39, no. 3&4: 720-729.

Papert, Seymour and Cynthia Solomon. 1971. *Twenty Things to Do With a Computer.* DSpace@MIT.

Papert, Seymour. 1993. *Mindstorms: Children, Computers, and Powerful Ideas*. 2nd ed.New York: Basic Books.

Paz, Taz. and Uri Leron. 2009. The Slippery Road from Actions on Objects to Functions and Variables. *Journal for Research in Mathematics Education* 40, no. 1: 18-39.

Pea, Roy. 1983. Chameleon in the Classroom: Developing Roles for Computers. Montreal, Canada, April, 1983.

Pea, Roy D. and D. Midian Kurland. 1984. On the Cognitive Effects of Learning Computer Programming. *New Ideas in Psychology* 2, no. 2: 137-168.

Ramsey, Fred L. 2002. *The Statistical Sleuth : A Course in Methods of Data Analysis*. Ed. Daniel W. Schafer. 2nd ed. ed.Australia; Pacific Grove, CA: Duxbury/Thomson Learning.

Rich, Peter, Keith Leatham, and Geoffrey Wright. 2012. Convergent Cognition. *Instructional Science*: 1-23.

Robertson, John. 1998. Paradise Lost: Children, Multimedia and the Myth of Interactivity. *Journal of Computer Assisted Learning* 14, no. 1: 31-39.

Schanzer, Emmanuel. 2011. Algebraic Functions, Computer Programming, and the Challenge of Transfer.

Schanzer, Emmanuel. Bootstrap. http://www.bootstrapworld.org (accessed 11/10, 2012).

Steele, Guy L., Jr. 2006. The History of Scheme. http://jaoo.dk/jaoo2006/speakers/show_speaker.jsp?oid=44 (accessed April 8, 2011).

Subramaniam, Venkat. 2008. *Programming Scala, Tackle Multicore Complexity on the Java Virtual Machine*. The Pragmatic Bookshelf.

Sutherland, Rosamund. 1991. Some Unanswered Research Questions on the Teaching and Learning of Algebra. *For the Learning of Mathematics* 11, no. 3: pp. 40-46.

Sutherland, Rosamund. 1989. Providing a Computer Based Framework for Algebraic Thinking. *Educational Studies in Mathematics* 20, no. 3: 317-344.

United States. National Mathematics Advisory Panel. 2008. *Foundations for Success the Final Report of the National Mathematics Advisory Panel.* Ed. United States. Dept. of Education.Washington, D.C.: U.S. Dept. of Education.

Wampler, Dean. 2011. *Functional Programming for Java Developers: Tools for Better Concurrency, Abstraction, and Agility*. 1st ed.O'Reilly Media, Inc.

Wright, Geoffrey A., Peter Rich, and Keith R. Leatham. 2012. How Programming Fits with Technology Education Curriculum.(report). *Technology and Engineering Teacher,* 3.

APPENDICES

APPENDIX A ASSESSMENTS

Vista Heights Pre Assessment

Name: ____

Date: ____

Age (circle one):

10 11 12 13 14

Gender (circle one):

Male Female

Grade Level (circle one):

6 7 8 9

Current Math Class (circle one):

Basic Math Pre-Algebra Algebra Geometry

Race/Ethnicity (circle all that apply)

Caucasian African American Asian Latino/a Native American

What are your reasons for participating in this class?

How do you feel about your programming skills?

1	2	3	4	5	6	7
I've never programmed		OK. I've done a little programming		Fairly confident. I've programmed several things, but would need help to remember		Very confident. I could program my own games.

Rate how quickly you learn new technologies?

1	2	3	4	5	6	7
Technology? I have a hard time learning new technologies.		I'm ok – but not too confident at learning technology.		Fairly confident. I can figure most technologies out.		Very confident. I'm amazing at technology.

What is technology (define technology)?

How do you feel about your math skills?

1	2	3	4	5	6	7
Not Good at math		OK. I'm a little below average at Math.		Fairly confident. I'm pretty good at math – better than most.		Very confident. I'm amazing at math.

Complete all the answers to the best of your knowledge

1. The following questions are about the expression:

$$3\underset{\uparrow}{n}-5$$

A. The arrow points to n. What does n stand for?
B. Could n represent 21?
C. Could n represent $(21+15)$?
D. Could n represent $(a+1)$? Why or why not?

2. Can you tell which is larger, $2n$ or $n+2$? Please explain your answer.

3. Solve the following expression (show your work): $5+1*4-8/2$

4. Consider the following number sentence: $n-84=227$
The value of n that makes this number sentence true is 311, because $311-84=227$

If 10 is added to each side of the number sentence, the new number sentence is as follows:
$n-84+10=227+10$

What value of n makes this new number sentence true?

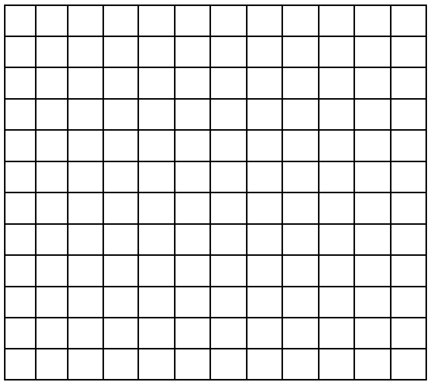
Explain how you got your answer.

5. A friend sees the word “function” in your math book and asks what it means. In terms of mathematics, how do you explain “function” to them?

6. Consider the following function:

$f(x) = {}^{3}\!/_{4}\, x + 2$

A. On the graph below, plot and label $f(0)$ and $f(4)$.



B. How far apart are the two points? Show your work.

7. Given $f(x) = 5(x+3)$, find $f(n+2)$.

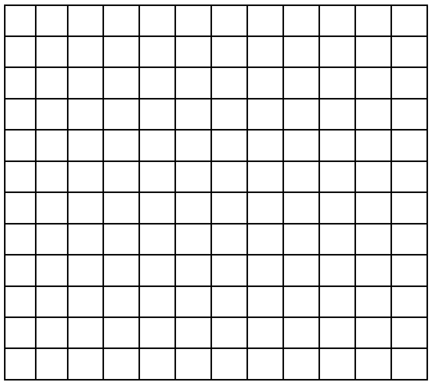
8. Using the following two functions, find the value of the composite function $f(g(5))$. Show your work.

$f(x) = 2x + 1$

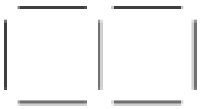
$g(y) = y^2$

9. Graph the following piecewise function

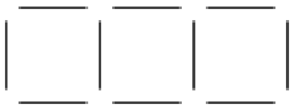
$$f(x)=\begin{cases} 2x-4 & \textit{for} \quad x<2 \\ -x+2 & \textit{for} \quad x\geq 2 \end{cases}$$



10. If a side of each square in the shape below is 1 toothpick, then it takes 7 toothpicks to make 2 squares in a row.



A. How many toothpicks are required to make 3 squares in a row?



B. How many toothpicks are required to make 7 squares in a row? Show your work.

C. Describe in words how you could figure out how many toothpicks are needed to build any number of squares in a row.

D. Joan built 30 squares in a row. How many toothpicks did she use? Show your work.

E. Tom built a row of squares using 151 toothpicks. How many squares did he build?

Vista Heights Post Assessment

Name: ______________________________

Date: ______________________________

Age (circle one):

10 11 12 13 14

Gender (circle one):

Male Female

Grade Level (circle one):

6 7 8 9

Current Math Class (circle one):

Basic Math Pre-Algebra Algebra Geometry

Race/Ethnicity (circle all that apply)

Caucasian African American Asian Latino/a Native American

What are your reasons for participating in this class?

How do you feel about your programming skills?

1	2	3	4	5	6	7
I've never programmed		OK. I've done a little programming		Fairly confident. I've programmed several things, but would need help to remember		Very confident. I could program my own games.

Rate how quickly you learn new technologies?

1	2	3	4	5	6	7
Technology? I have a hard time learning new technologies.		I'm ok – but not too confident at learning technology.		Fairly confident. I can figure most technologies out.		Very confident. I'm amazing at technology.

What is technology (define technology)?

How do you feel about your math skills?

1	2	3	4	5	6	7
Not Good at math		OK. I'm a little below average at Math.		Fairly confident. I'm pretty good at math – better than most.		Very confident. I'm amazing at math.

Complete all the answers to the best of your knowledge

1. The following questions about the following expression:

$$7\underset{\uparrow}{x}+5$$

A. The arrow points to x. What does x mean?

B. Could x represent 217? Why or why not?

C. Could x represent $(217+15)$? Why or why not?

D. Could x represent $(n+5)$? Why or why not?

2. Which is larger, $4a$ or $a+4$? Please explain your answer.

3. Solve the following expression (show your work): $3+5*3-9/3$

4. Consider the following number sentence: $n-80=311$
The value of n that makes this number sentence true is 391, because $391-80=331$.

If 10 is added to each side of the number sentence, the new number sentence is as follows:
$n-80+10=311+10$

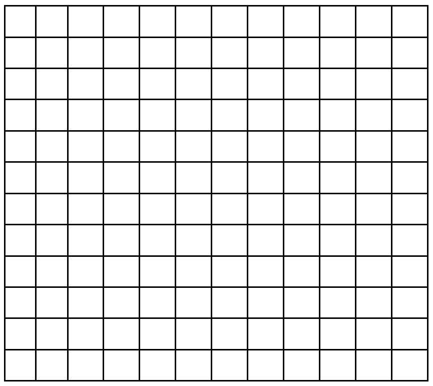
What value of n makes this new number sentence true?
Please explain how you got your answer.

5. A friend sees the word "function" in your math book and asks what it means. In terms of mathematics, how do you explain "function" to them?

6. Consider the following function:

$f(x) = {}^4\!/_3\,x - 1$

A. On the graph below, plot and label $f(0)$ and $f(3)$.



B. How far apart are the two points? Show your work.

7. Given $f(x) = 5(x+3)$, find $f(5-n)$.

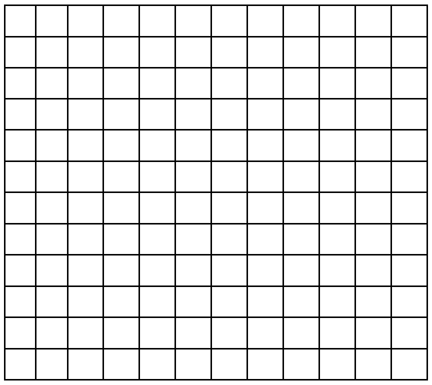
8. Using the following two functions, find the value of the composite function, $f(g(5))$. Show your work.

$f(x) = 2x$

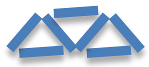
$g(y) = y^2$

9. Graph the following piecewise function

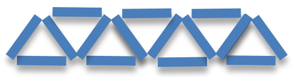
$$f(x)=\begin{cases} \frac{3}{2}x+2 & \text{for } x<2 \\ -\frac{1}{2}x+2 & \text{for } x\geq 2 \end{cases}$$



10. If a side of each triangle in the shape below is 1 toothpick, then it takes 7 toothpicks to make 3 triangles in a row.



A. How many toothpicks are required to make 7 triangles in a row?



B. How many toothpicks are required to make 20 triangles in a row? Show your work.

C. Write a function to demonstrate how you could figure out how many toothpicks are needed to build any number of triangles in a row.

D. Joan built 50 triangles in a row. How many toothpicks did she use? Show your work.

E. Tom built a row of triangles using 457 toothpicks. How many triangles did he build?

Lehi Pre Assessment

Name: ____

Date: ____

Age (circle one):

10 11 12 13 14

Gender (circle one):

Male Female

Grade Level (circle one):

6 7 8 9

Current Math Class (circle one):

Basic Math Pre-Algebra Algebra Geometry

Race/Ethnicity (circle all that apply)

Caucasian African American Asian Latino/a Native American

What are your reasons for participating in this class?

How do you feel about your programming skills?

1	2	3	4	5	6	7
I've never programmed		OK. I've done a little programming		Fairly confident. I've programmed several things, but would need help to remember		Very confident. I could program my own games.

Rate how quickly you learn new technologies?

1	2	3	4	5	6	7
Technology? I have a hard time learning new technologies.		I'm ok – but not too confident at learning technology.		Fairly confident. I can figure most technologies out.		Very confident. I'm amazing at technology.

What is technology (define technology)?

How do you feel about your math skills?

1	2	3	4	5	6	7
Not Good at math		OK. I'm a little below average at Math.		Fairly confident. I'm pretty good at math – better than most.		Very confident. I'm amazing at math.

Complete all the answers to the best of your knowledge

1. The following questions are about the expression:

$$3\underset{\uparrow}{n}-5$$

A. The arrow points to n. What does n mean?

B. Could n represent 21? Why or why not?

C. Could n represent $(21+15)$? Why or why not?

D. Could n represent $(a+1)$? Why or why not?

2. The following questions are about the expression:

$$3+4\underset{\uparrow}{=}7$$

A. The arrow above points to the = symbol. What is the name of symbol?

B. What does the symbol mean?

C. Can the symbol mean anything else? If yes, please explain.

3. Which is larger, $2n$ or $n+2$? Please explain your answer.

4. Solve the following expression (show your work): $5+1*4-8/2$

5. Consider the following number sentence: $n - 84 = 227$
The value of n that makes this number sentence true is 311, because $311 - 84 = 227$

If 10 is added to each side of the number sentence, the new number sentence is as follows:
$n - 84 + 10 = 227 + 10$

What value of n makes this new number sentence true?
Please explain how you got your answer.

6. A friend sees the word "function" in your math book and asks what it means. In terms of mathematics, how do you explain "function" to them?

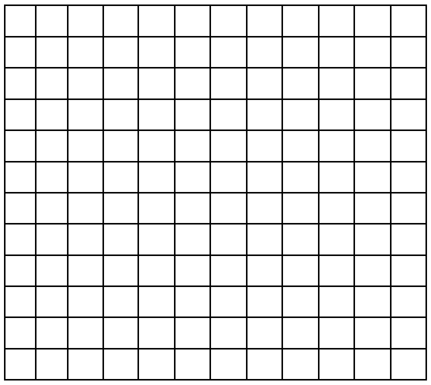
7. A function can be written in computer code (such as WeScheme) or as algebra notation. Below are two functions named *f* and *g*. Each function takes one parameter. Find the value of the composite function listed, using whichever syntax is most familiar to you. Show your work.

Computer code	Algebra
(define (f x) (+ (* 2 x) 1))	$f(x) = 2x + 1$
(define (g y) (/ y 3))	$g(y) = y/3$
Find the value: (f (g 6))	Find the value: $f(g(6))$

8. Find the values for the following function, using the syntax which is most familiar to you.

Computer code	Algebra
(define (f x) (+ (* (/ 3 4) x) 2))	$f(x) = {}^{3}\!/_{4}\,x + 2$
Find $(f\ 0)$ and $(f\ 4)$	Find $f(0)$ and $f(4)$

A. On the graph below, plot and label.



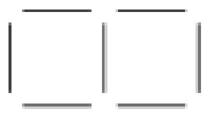
B. Using the Pythagorean Theorum ($a^2 + b^2 = c^2$), find how far apart are the two points? Show your work.

9. Given $f(x) = 5(x+3)$, find $f(n+2)$.

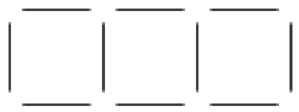
10. Using the syntax you prefer, solve the function for the value 5.

Computer code	Algebra
(define (f x) (cond [(< x 2) (- (* 2 x) 4)] [(>= x 2) (+ x 2)]))	$f(x) = \begin{cases} 2x-4 & for \quad x < 2 \\ x+2 & for \quad x \geq 2 \end{cases}$
Find the value: $(f\ 5)$	Find the value: $f(5)$

11. If a side of each square in the shape below is 1 toothpick, then it takes 7 toothpicks to make 2 squares in a row.



A. How many toothpicks are required to make 3 squares in a row?



B. How many toothpicks are required to make 7 squares in a row? Show your work.

C. Describe in words how you could figure out how many toothpicks are needed to build any number of squares in a row.

D. Joan built 30 squares in a row. How many toothpicks did she use? Show your work.

E. Tom built a row of squares using 151 toothpicks. How many squares did he build?

Lehi Post Assessment

Name: ______________________________

Date: ______________________________

Age (circle one):

10 11 12 13 14

Gender (circle one):

Male Female

Grade Level (circle one):

6 7 8 9

Current Math Class (circle one):

Basic Math Pre-Algebra Algebra Geometry

Race/Ethnicity (circle all that apply)

Caucasian African American Asian Latino/a Native American

What are your reasons for participating in this class?

How do you feel about your programming skills?

1	2	3	4	5	6	7
I've never programmed		OK. I've done a little programming		Fairly confident. I've programmed several things, but would need help to remember		Very confident. I could program my own games.

Rate how quickly you learn new technologies?

1	2	3	4	5	6	7
Technology? I have a hard time learning new technologies.		I'm ok – but not too confident at learning technology.		Fairly confident. I can figure most technologies out.		Very confident. I'm amazing at technology.

What is technology (define technology)?

How do you feel about your math skills?

1	2	3	4	5	6	7
Not Good at math		OK. I'm a little below average at Math.		Fairly confident. I'm pretty good at math – better than most.		Very confident. I'm amazing at math.

Complete all the answers to the best of your knowledge

1. The following questions are about the following expression:

$$7\underset{\uparrow}{x}+5$$

A. The arrow points to x. What does x mean?

B. Could x represent 217? Why or why not?

C. Could x represent $(217+15)$? Why or why not?

D. Could x represent $(n+5)$? Why or why not?

2. The following questions are about the expression:

$$8-3\underset{\uparrow}{=}7$$

A. The arrow above points to the = symbol. What is the name of symbol?

B. What does the symbol mean?

C. Can the symbol mean anything else? If yes, please explain.

3. Which is larger, $4a$ or $a+4$? Please explain your answer.

4. Solve the following expression (show your work): $3+5*3-9/3$

5. Consider the following number sentence: $n-80=311$
The value of n that makes this number sentence true is 391, because $391-80=331$.

If 10 is added to each side of the number sentence, the new number sentence is as follows:
$n-80+10=311+10$

What value of n makes this new number sentence true?
Please explain how you got your answer.

6. A friend sees the word “function” in your math book and asks what it means. In terms of mathematics, how do you explain “function” to them?

7. Below is a function written in the Racket programming language. The name of the function is *f*, and it takes one parameter called *x*.

```
(def (f x) ( −( * ( ÷4 3 ) x) 1))
```

The same function can be written in algebra notation.

$$f(x)={}^{4}\!/_{3}\,x-1$$

A. On the graph below, plot and label $f(0)$ and $f(3)$.

B. Using the Pythagorean Theorum ($a^2+b^2=c^2$), find how far apart are the two points? Show your work.

7. Given $f(x) = 5(x+3)$, find $f(5-n)$.

8. Using the following two functions, find the value of the composite function, $f(g(5))$. Show your work.

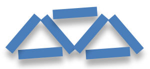
$f(x) = 2x$

$g(y) = y^2$

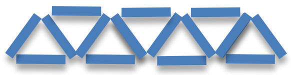
9. Graph the following piecewise function

$$f(x) = \begin{cases} \frac{3}{2}x + 2 & for \quad x < 2 \\ -\frac{1}{2}x + 2 & for \quad x \geq 2 \end{cases}$$

10. If a side of each triangle in the shape below is 1 toothpick, then it takes 7 toothpicks to make 3 triangles in a row.



A. How many toothpicks are required to make 7 triangles in a row?



B. How many toothpicks are required to make 20 triangles in a row? Show your work.

C. Write a function to demonstrate how you could figure out how many toothpicks are needed to build any number of triangles in a row.

D. Joan built 50 triangles in a row. How many toothpicks did she use? Show your work.

E. Tom built a row of triangles using 457 toothpicks. How many triangles did he build?

APPENDIX B ASSESSMENT GRADING RUBRIC

Variable: **What is n… 3n-5**

A)	NS	
B)	1pt	for saying “yes”
C)	1pt	.5 for saying “yes”, .5 for elaboration
D)	1pt	.5 for saying “yes”, .5 for elaboration

Variable: **Can you tell which is larger? 2n or n+2?**

1pt	2n IF followed by example where $n>2$
1pt	n+2 IF followed by example where $n<2$
2pts	for saying it depends and giving correct examples
3pts	for giving three cases and accompanying conditions (n being <, >, or = to 2)

Other: 3+4=7

A)	.5pt	
B)	.5pt	for stating in words that 3 plus 4 equals 7
C)	1pt	for something like assignment

Order-of-Operations: **8+6*2/3-(-3)**

1pt	correct answer (15)
1pt	correct work using order-of-operations

Other: n-84+10=227+10 What value of n makes this true?

1pt	correct answer (311)
1pt	for either algebraic manipulation to isolate n

Transfer: Plot and determine distance between two points

A)	1pt	drawing or labeling x- and y-axis or origin
	1pt	correctly orienting (2,1)
	1pt	correctly orienting (-1,-3)
B)	2pts	solving via Pythagorean Theorem $a^2+b^2=c^2$
	1pt	correct answer (5)
OR		
	2pts	solving via distance formula d=(x2-x1)2+(y2-y1)2
	1pt	correct answer (5)

Function: How do you explain "function"?

1pt	reference to input output
2pts	correct elaboration of the "machine"
1pt	exceptionally impressive answer

Function: Given $f(x)=5(x+3)$, find $f(n+2)$.

1pt	showed composition
2pts	correct answer – f(n-2)=5((n-2)+3)

Function: Given $f(x)=2x+1$ and $g(y)=y^2$, find $f(g(5))$.

1pt	applied 5 to g(y)
2pts	correct answer – (f(g(y))=2(25)+1=51

Transfer: **graph** $f(x)=\begin{cases} 2x-4 & for \quad x<2 \\ -x+2 & for \quad x\geq 2 \end{cases}$

1pt	correct for x<2

1pt	correct for x>= 2
1pt	correctly drawn graph

Toothpicks

A)	**NS**	***This isn't really a question. If they miss this they're in sorry shape.***
B)	5pt	correct answer
C)	1pt	suggests building a function
	1pt	explain why a function is applicable
	1pt	actually build a function
	1pt	function is correct (variation of 3n+1)
	1pt	explain how function was engineered
D)	1pt	correctly plugs 30 into function (e.g. 3(30)+1)
	1pt	answers correctly (91)
	no points will be awarded if solution is derived strictly from drawing a picture	
E)	1pt	plugging in or manipulating function correctly
	1pt	correct answer (50)

APPENDIX C STATISTICAL ANALYSIS CHARTS

Multiple Regression Statistical Analysis for Bootstrap Course at Vista Height Middle School (N=26)

Category	***t-ratio***	***Prob |t|***	***Lower 95%***	***Upper 95%***
Variables	3.81	.0009	1.00	3.38
Functions	5.86	<.0001	3.25	6.80
Transfer	1.29	.21	.25	1.06
PEMDAS	.07	.94	3.25	6.80
Other	2.49	.02	.40	4.96

SMD Effect Size Statistical Analysis for Bootstrap Course at Vista Heights Middle School (N=26)

Category	***Mean for Treatment Group***	***SD for Treatment Group***	***Mean for Control Group***	***SD for Control Group***	***SD Pooled***	***Effect Size***
Variables	1.44	1.65	-.06	1.66	1.65	.91
Functions	1.17	3.39	-1.77	2.67	5.43	.54
Transfer	.67	1.41	-.14	.60	1.91	.43
PEMDAS	.11	.65	.26	.71	1.28	-.12
Other	.44	5.59	.32	3.99	7.94	.02

Multiple Regression Statistical Analysis for Bootstrap Course at Lehi High School (N=16)

Category	*t-ratio*	*Prob |t|*	*Lower 95%*	*Upper 95%*
Variables	-3.76	.0024	1.7	.46
Functions	1.84	.09	-.39	1.11
Transfer	1.48	.15	-.32	1.69
PEMDAS	0	1	-.70	.70
Other	3.94	.0017	1.31	4.5

SMD Statistical Analysis for Bootstrap Course at Lehi School (N=16)

Category	*Mean for Treatment Group*	*SD for Treatment Group*	*Mean for Control Group*	*SD for Control Group*	*SD Pooled*	*Effect Size*
Variables	1.43	2.09	-.05	1.04	1.57	.93
Functions	.86	3.67	-.56	1.63	2.70	.52
Transfer	1	1.15	.22	.62	.89	.88
PEMDAS	1.21	.91	0	1	.96	1.26
Other	.64	2.12	-2.17	2.38	2.27	1.24

APPENDIX D INTERVIEW SCRIPT

Write the following Racket code on the whiteboard:

```
(define (times3 value)
  (* 3 value))
```

Question: What does this mean?

Leave the Racket code on the board. Write the following Racket functions, each time asking the question.

How does the computer solve:

`(times3 1)`

`(times3 5)`

`(times3 0)`

`(times3 -2)`

Refer back to the Racket code that defines times3.

Question: What is 'value'?

Question: What does that mean?

Questions: How does it relate to a function?

Write the following mathematical function on the board.

times3(value) = 3value

Question: Where have you seen a function written like this one?

Question: What does it mean?

Write the following mathematical function on the board.

f(x)=3x

Question: How do you recognize the function now?

Question: What is the difference between f(x) and the Racket function times3?

Question: Show me other ways you can represent times3?

Hand the student some graph paper.

Question: Draw f(x) on the graph paper.

Question: What does value or x mean here?

Write the following algebraic functions on the board.

f(x)=x+5
g(y)=10-y
f(g(x))

Question: What does f(g(x)) mean?
Question: How do you write f(x) and g(y) in Bootstrap?

Question: Show me how to combine the two functions into one function.

Question: How is this function you just wrote different from f(g(x))?

Question: What is a function?

Question: What is a variable?

Question: How do functions and variables work together?

APPENDIX E INTERVIEWS

Dixon Middle School

Interviewee Information
Name: Pedro
Age: 14
Grade: 8
Current Math: Algebra I

Interview Context
Location: Dixon Middle School
Date: May 17, 2011
Time: 3:00 p.m.
Interview time: 25 minutes

	Interviewer:	
1	So you about to finish Algebra I?	
2	**Pedro:** *Head nod*	
	Interviewer:	
3	Getting A's? B's, sorta, A's and B's?	
4	**Pedro:** *Head nod*	
	Interviewer:	
5	Alright, I've got a daughter who is in algebra I. Actually, it is the second time she has taken it. She got C's all along and said, "I've got to do this again."	
	So, that's great.	
6	Alright, so the idea here is, uhm, we just want to know if the stuff you are learning in Bootstrap means anything in math. And so I'm just going to ask you	

	some Bootstrap questions and some math questions. And you go ahead. You write on the board here, and uh, you go ahead and tell me, write out your answers. Just talk to me as you are going.	
7	Is that ok?	
8	**Pedro:** *Head nod*	
	Interviewer:	
9	Ok, so let's do. Here we are.	
10	Okay, question 1.	
11	I'll write my stuff in red, and you can write yours in green.	
12	So I'll give you the good marker.	
13	Okay, so we are going to say	
	(*Write the following on the board*)	
14	(define (times3 value) (* 3 value))	
15	Tell me, what does this mean?	
	(*pause*)	
16	What does this mean?	
	Pedro:	
17	"I don't know"	17 Being shy
	Interviewer:	
18	What will the computer, how will, what will the computer do with this? This is in the Racket stuff you've been doing. What would the computer do with this? Particularly, what if I wrote?	
	(*Write the following on the board*)	
19	(times3 1)	
20	How would that get evaluated?	
	Pedro:	
21	(*Confused stare*)	

22	Or, what would this return?	
	Pedro:	
23	(*Confused stare*)	
24	You want to try it on a computer?	
	Pedro:	
25	(*Confused stare*)	
	Interviewer:	
26	Let's try it over here and see what it comes up with.	
	(*Move over to the computer, have Pedro in driver's seat*)	
27	Let' just do it down here (*referring to the evaluation window*) for right now.	
28	So just type in define space paren times 3 no space on 3, just times 3 space value	
29	then close that	
	and then enter and then open paran star which is times space 3 space value close paran paren	
30	go ahead and hit enter	
31	ok, go ahead and type in times3 space 1	
32	What do you get back?	
	Pedro:	
33	3	33 realizing
	Interviewer:	
34	Alright, now lets do the same thing but put in 5	
35	15, ok,	35 understanding
36	now let's come back over here	
	(*move back to whiteboard*)	
37	So, what if we did times3 0, what would I get	

	Pedro:	
38	0	38 Showing confidence
	Interviewer:	
39	You get 0, alright	
40	And what if I put in -2	
	Pedro:	
41	-6	41 Showing confidence
	Interviewer:	
42	Right. So is this kind of hard to read?	
43	Ok, so it is my handwriting that is slowing us up?	
44	What is value?	
	Pedro:	
45	A number.	45 explaining
	Interviewer:	
46	Ok, uhm	
47	ok so values a number uhm,	
48	how does it relate to the function?	
	Pedro:	
	(*blank confused stare*)	
	Interviewer:	
49	Don't know, ok. What does value do in this function?	
	Pedro:	
50	It's times	50 Understanding
	Interviewer:	
51	Ok, so it's what you times. Is that what you are saying?	
	Pedro:	
52	(*head nod*)	
	Interviewer:	
53	ok, alright,uhm	

54	Have you ever heard this called a variable?	
	Pedro:	
55	yes	55 answering
	Interviewer:	
56	So value here is a variable.	
57	Let me, what if I wrote it like	
	(*write on board*)	
58	times3(value)=3value	
59	Have you seen anything written like that?	
	Pedro:	
60	(*head nod*)	60 recognizing
	Interviewer:	
61	Where have you seen something written like that?	
	Pedro:	
62	In math	62 recognizing
	Interviewer:	
63	In math, ok. Do you know another way it might be written?	
	Pedro:	
	(*write on board*)	
64	3(x)=3x	64 demonstrating
	Interviewer:	
65	Now let's start having some fun with that.	
66	Tell me, what's the difference between this 3 x equals 3 x and this up here	
	(*pointing to function*)	
	Pedro:	
67	(*confused stare*)	

	Interviewer:	
68	That's fine	
69	Is there any difference between this and this?	
	Pedro:	
70	No	70 understanding
	Interviewer:	
71	Why not?	
	Pedro:	
72	Same thing	72 understanding
	Interviewer:	
73	Yah, ok.	
74	Lets to this on this paper	
	(*hand piece of graph paper and pencil to Pedro*)	
75	Can you draw this function right here?	
	Pedro:	
76	*Writes f(x)=3x*	
	Interviewer:	
77	Can you draw it as a graph?	
	Pedro:	
78	*Draws two axes. Calculate three points on the graph, draws line through the points.*	*78 demonstrating*
	Interviewer:	
79	Great. Alright. What does x mean in this? What is x?	
	Pedro:	
80	A value	
	Interviewer:	
81	Great. Alright. I'm going to write something again, I'm going to make it be red.	

	On board write	
82	(define (doMoreMath value)	
	(cond	
	[(< value 3) (+ 2 value)]	
	[(>= value 3) (* 3 value)]))	
83	Ok, can you draw this as a graph on your paper?	
84	Does that make sense? Is my handwriting too bad?	
85	*long pause*	
86	Does it make any sense, or… Ok, that's fine, that's fine	
87	Have you seen anything like that in math?	
	Pedro:	
88	No	
	Interviewer:	
89	Ok, great.	
90	Let's do this then.	
	draw on board	
91	f(x) = x + 5	
	g(y) = 10 – y	
92	So back to math, alright	
93	What would f(g(x)) mean?	
	Pedro:	
94	*confused stare*	
	Interviewer:	
95	Doesn't make sense?	
96	Ok, so, lets go ahead, and how would we write this one, f(x) in bootstrap.	
97	*write on board (define*	
98	What should we call it?	

	Pedro:	
99	*shrug*	
	Interviewer:	
100	plus5?	
101	What comes next? Maybe something like x?	
	Pedro:	
102	*nod*	
	Interviewer:	
103	Acutally I've got to put a paren right here.	
	What would I put here?	
	Pedro:	
104	5	104 following
	Interviewer:	
105	a 5	
	Pedro:	
106	yes	106 following
	Interviewer:	
107	what else	
	Pedro:	
108	plus	108 following
	Interviewer:	
109	Now let's do this one in Bootstrap	
	Write on board (define	
110	Let's do y this time	
111	What do I put here?	
	Pedro:	
112	minus	112 interacting
	Interviewer:	
113	minus and what else	

	Pedro:	
114	5	114 following
	Interviewer:	
115	5	
116	Now, How could you? Let’s see here. So if you said, plus5 5, what do I get?	
	Pedro:	
117	10	117 understanding
	Interviewer:	
118	And if we say 10 minus 5 what do we get	
	Pedro:	
119	5	119 following
	Interviewer:	
120	Now, what would I do if I wanted to say plus 5 and then, and then instead of saying 5 here I would say 10 minus and 5. How would I write that?	
	Pedro:	
121	*confused look*	
	Interviewer:	
122	Do you understand what I am saying?	
	Pedro:	
123	No	123 Being confused
	Interviewer:	
124	Ok, so. The value of this one is 5. So I want to say plus 5 and I want to put in here 10 minus. Can I do that?	
	Pedro:	
125	Yes.	125 understanding
	Interviewer:	
126	And a 5. Would that be correct?	
	Pedro:	
127	Yes	127 following

	Interviewer:	
128	Now, does this make any more sense?	
129	What does it mean?	
	Pedro:	
130	*confused look*	
	Interviewer:	
131	It's alright, if you really can't explain that? Makes kinda more sense now?	
	Pedro:	
132	yes	132 Pleasing interviewer
	Interviewer:	
133	Alright. Let me ask just two more, just three more questions. We don't have to do anything on the board.	
134	What is a function?	
	Pedro:	
135	*Confused look*	
	Interviewer:	
136	It's ok.	
137	What's a variable?	
	Pedro:	
138	A number.	138 answering
	Interviewer:	
139	And how do functions and variables relate to each other? How do they work together?	
	Pedro:	
140	*shrugs*	
	Interviewer:	
141	Ok, that's perfectly fine Pedro. You've been a great help. I appreciate it.	
142	Is English your first language? Or is Spanish…	

Coding
33 realizing
60 recognizing
62 recognizing
64 demonstrating
78 demonstrating

<u>Shy or confused</u>
17 Being shy
123 Being confused
132 Pleasing interviewer

<u>working at level</u>
35 understanding
50 Understanding
70 understanding
72 understanding
117 understanding
125 understanding
104 following
106 following
108 following
114 following
119 following
127 following
38 Showing confidence
41 Showing confidence
45 explaining
112 interacting

<u>Coginitive bridge</u>
33 realizing
60 recognizing
62 recognizing
64 demonstrating
78 demonstrating
138 answering
55 answering

Statements	Category	Property	Hypothesis
Being Shy	Shy or Confused		Quiet person
Being Confused			
Pleasing interviewer			

Statements	Category	Property	Hypothesis
understanding	Working at Level	Understanding discussion	Coherent, answers when he knows
Following interacting		Follow logical steps	Able to follow logical steps
Showing confidence explaining		Able to do work at skill level	Answering on his own

Statements	Category	Property	Hypothesis
recognizing	Cognitive bridge		Making connection
demonstrating			
answering			Understood, follow logic
realizing		Made discovery	Programming experience allowed him to recognize math syntax

3	33 realizing	
(*head nod*)	60 recognizing	
In math	62 recognizing	
3(x)=3x	64 demonstrating	
Draws two axes. Calculate three points on the graph, draws line through the points.	*78 demonstrating*	
A number.	138 answering	
yes	55 answering	

Finding: Was able to make connection between functions and math function syntax, but was not able to go beyond in the time of the interview.

Interviewee Information
Name: Tyler
Age: 13
Grade: 7
Current Math: Pre-Algebra A

Interview Context
Location: Dixon Middle School
Date: May 17, 2011
Time: 3:30 p.m.
Interview time: 25 minutes

	Purpose: evaluate understanding of algebra due to Bootstrap course.	
	Context: Tyler has been regular for several weeks, the course will be finishing in a few days	
	On board is written	
	(define (times3 value)	
	(* 3 value))	
	Interviewer	
1	Tell me, what does this mean?	
	Tyler	
2	It's saying times 3 of a value of something, I think. And it's defining what it is. And then it's kind of like a math problem, I guess. Like times 3 value, like maybe you have the value is an orange. You times that 3 times, and you have like maybe three oranges. Or depending on how many values there are. You have so many.	2 Relating to math 2 Understanding Racket 2 Giving example
	Interviewer	
3	Uh huh. That's a good explanation. So what is value?	

	Tyler	
4	Value could be like really anything that you wanted it to be. Because it's just like in the problems when you have just like a letter. There something that represents something, and something that you have to find out or you don't really need to figure out, it's just there.	4 Explaining value En vivo
	Interviewer	
5	Ok. Something just there.	
	Tyler	
6	yea.	
	Interviewer	
7	How does that relate to a function?	
	Tyler	
8	Function?. That like it has multiple options of what it could be, so like, it could have multiple different functions, depending on what you give it.	8 Fumbling answer
	Interviewer	
9	Ok. I'm supposed to write in code, I keep forgetting that. So I can write pretty quick.	
10	Ok, What does… this mean	
	write on board	
11	(times3 1)	
	Tyler	
12	You are timesing 3 by 1, so it's like 3 times 1.	12 Relating to math
	Interviewer	
13	Ok. What if I put in here 5.	

	Tyler	
14	Then it just be like 1 … 5, be like 3 times 5, but the computer reads it differently. Like addition and subtraction or that thing in first, and then put the other two numbers with the space in between them.	14 Relating to math. Expanding answer to difference between math and Racket
	Interviewer	
15	uhm hm.	
	Tyler	
16	If or, like that.. or so it reads it that way. But we'd read a problem like that 3 times 5.	16 Relating to math
	Interviewer	
17	What about 0.	
	Tyler	
18	Three times-ing it by 0 and it. Each time the value changes you get a different answer. And sometimes the answer is big. Sometimes it is small.	18 Moving beyond questioning to explain point
	Interviewer	
19	uhm hm.	
	Tyler	
20	But it is the same problem with the one different number changing.	20 Clarifying
	Interviewer	
21	What if I said, something like that?	
22	*Write -2*	
23	Is that alright.	
	Tyler	
24	Yeah.	
	Interviewer	
25	Yep.	
	Tyler	
26	Because the value can be like any number or …	

	Interviewer	
27	Alright. Now, what if I wrote	
	write on board	
28	time3(value) = 3value	
29	Where have you seen something like that?	
	Tyler	
29	*shrug*	*29 Not understanding*
	Interviewer	
30	What if I said	
	write on board	
31	f(x) = 3 * x	
32	Where have you seen that?	
	Tyler	
33	What does the f stand for? Or is f just a letter?	33 Not understanding
	Interviewer	
34	uhm hm.	
	Tyler	
35	Uhm in math… Oh, the problem is like you ah. When I normally see this, it's normally like there is like a letter here, and then normally like a letter, and a sign and another number, then you times this by those, and then you get that, equals like this, and then you sort of divide from each side like …	35 Relating to experienced math 35 Realizing similarities
	Interviewer	
36	ok. So you are saying something like this	
	write on board	
37	y=3x	

	Tyler	
38	Then it would be like saying, that like y equals whatever 3 times x equals and so, then if you, and then you have to figure out what x equals in order to find out what y equals because they equal each other. Well, this problem equals something that equals y.	38 Showing math understanding
	Interviewer	
39	Ok. Now, this syntax here actually, is usually done like that.	
	write on board	
40	f(x)=3x	
41	this something here is meaning, you know we just use f because it's there, the function that takes x is this.	
42	And so, is this different… How is this different from this?	
	pointing to time3 Racket function and f(x)=3x	
	Tyler	
43	define is like stating something and a function is like giving it an option of sort.	43 Fumbling answer
	Interviewer	
44	uhm hum.	
45	And then, which one, ok so this is like stating something	
	Tyler	
46	yeah	
	Interviewer	
47	and then this	
	Tyler	
48	is like giving something a function for like, say,	
49	you were to finding like a key but when you gave that key a function of when you hit it like something happened like maybe you went up or depending upon what button or something.	49 Exploring the concept of function
	Interviewer	
50	Ok, great, uhm. Alright, let's have some more fun. Let's go ahead, right over here, oh,	

55	Can you tell me other ways to represent that? *pointing to f(x)=3x*	
	Tyler	
56	Like in coding or?	56 clarifying
	Interviewer	
57	Whatever you want.	
	Tyler	
58	Not that I can really think of.	
	Interviewer	
59	That is perfectly fine. Uhm, let's do. Have you done graphing.	
	Tyler	
60	Yeah, I've done graphing.	
	Interviewer	
61	Do you have a pencil?	
	Tyler	
62	I have a pen. Will that work?	
	Interviewer	
63	That will work. Ok. Can you graph that right there. *pointing to f(x)=3x*	
	Tyler	
64	I can't.	
	Interviewer	
65	Ok. noooooo problem. Let's go over here. Alright. Here's the big one.	
	write on board	
66	(define (doMoreMath value)	
	(cond	
	[(< value 3) (+ 2 value)]	
	[(>= value 3) (* 3 value)]))	
67	Alright. Can you show me that on a graph?	

	Tyler	
68	*nods no*	
	Interviewer	
69	Nooo. That's ok, not on a graph.	
	and	
70	Let's put this away before I paint myself *referring to whiteboard marker*	
71	Alright, let go to this problem.	
	write on board	
72	f(x) = x + 5	
	g(y) = 10 – y	
73	Alright, does that make any sense?	
	Tyler	
74	kind of.	
	Interviewer	
75	Explain to me what those mean.	
	Tyler	
76	Like, this f would maybe equal function and then that would be like f timesing equals whatever x is plus 5	76 Fumbling answer
77	and kind of the same down here, except that g probably stands for something else, and then, or that x is like another other number, and then since it is in parenthesis it means times, and that's also a letter equaling 10 minus whatever a number.	
	Interviewer	
78	Ok. So let me tell you. Let's say, all this means this is defining a function called f, and this parens here means f takes just one, what would you call that.	
	Tyler	
79	One what?	79 clarifying
	Interviewer	
80	Takes just one what would you call?	
	Tyler	
81	Number?	81 guessing

	Interviewer	
82	Or, what would you call x?	
	Tyler	
83	value?	83 guessing
	Interviewer	
84	Ok, one variable.	
	Alright, so we have this function, we are calling it x. Remember over here I called it times and I put in value. Same thing.	
85	So you have function x and then that is what the function is. It is kind of like taking the y equals but, it's something you do here. Now, and see here is just another name, another name for the variable.	
86	These two are just functions in math.	
	Tyler	
87	Ok	
	Interviewer	
88	Ok. Just like this function here is a, this function here is the same thing as this. Ok? They are just functions. Ok. uhm. Would.	
89	Did that make any sense? Or what does that mean?	
	Tyler	
90	Kind of the same in a sense. And it like, it was like defining our function, and then that's like a random number and that's a random number, but you're doing the function like timesing this but when you get that you are timesing it by that.	90 recognizing
	Interviewer	
91	Ok. Ok.	
92	Remember what I said, how this is kind of like, up here, we're just saying define, and we say, do more math, so that's just the name, and then this is just the variable. This is just not really times, its just saying, this is its notation. It doesn't mean times, it just means, it's saying the same thing as, oh, doMoreMath passing value.	

94		
	Tyler	
95	Oh, ok.	
	Interviewer	
96	So, if that's the case, does this make more sense? Where it is not really saying times, it's just saying this is the function that takes x.	
	Tyler	
97	Yeah.	
	Interviewer	
98	Then, what would this mean?	
	Writing	
99	f(g(x))	
	Tyler	
100	That, this is the function that takes x, but this function takes that one, to that one?	100 fumbling
	Interviewer	
101	Ok.	
102	Alright.	
103	How would you write these two in Bootstrap. Can you do it?	
	Tyler	
104	I think so.	104 hesitating
105	Uhm.	
	Interviewer	
106	You can even use the names f and g, it doesn't matter.	
	Tyler	
107	Ok, I don't know how to …, I don't know how I would like write the equal sign.	107 clarifying
108	That's just like enter for me.	
	Interviewer	
109	Uhm, hum. So don't worry about it.	
	Tyler	
110	Ok.	

	Interviewer	
111	Remember this is just the name, and this is the variable.	
	Tyler	
112	Okay.	
	writes	
113	(+ (f(x) 5))	113 fumbling
114	Is that right?	114 Pleasing interviewer
	Interviewer	
115	Let do it this way. define, let's say, this may make more sense. Define f and x.	
	Does that make more sense?	
	Tyler	
116	yeah	
	Interviewer	
117	define f and x, and that's just going to be times, I'm sorry, plus x and 5. And with this just say define g and y to be minus 10 and y.	
118	Seeing those two, do these make more sense?	
	Tyler	
119	Yeah.	
	Interviewer	
120	How do they make more sense?	
	Tyler	
121	It's like saying these are like uh that is the function that's like the variable of the function, but then it's telling you the main problem down here and it's like that the variable option plus like the function or something?	121 Realizing, Asking for help
	Interviewer	
122	uhm, hum.	
	Tyler	
123	of it. And then, that prob uhm not putting it all in one?	

	Interviewer	
124	uhm, hum. Ok. Is this different from this? *Referring to f(x) and Bootstrap*	
	(define (f x))	
	Tyler	
125	No, they are the same thing, just in different form?	125 realizing
	Interviewer	
126	Ok.	
127	So, now, if I said. … I wanted to call, and I said. And I want to call, f, and I wanted to pass it 6, what would happen?	
	Tyler	
128	Well, uhm, there would be… And it would be showing you what that function option…	128 fumbling
	Interviewer	
129	So, what would the answer be?	
	Tyler	
130	uhm?	
	Interviewer	
131	What would this return?	
	Tyler	
132	Like not counting this part.	132 clarifying
	Interviewer	
133	Well yeah, in other words, we've written this and we've written this. What would the computer give me now?	
	Tyler	
134	I'm not exactly sure.	
	Interviewer	
135	Alright.	
	So if we took, …. 6 and the name of the function is f. …	
	That would be	

	Tyler	
136	11	136 understanding
	Interviewer	
137	Now let’s get really tricky.	
	Tyler	
138	Ok.	
	Interviewer	
139	You ready?	
140	What if I said f instead of 6, I want to pass in g of 6. Can I do that?	
	Tyler	
141	It’s like, that, be like, takes like x, so that would be like saying another one so there would be 6 so there would be 12 no 11 and then	141 guessing
	Interviewer	
142	remember, g is over here.	
	Tyler	
143	Oh, that’s giving y function, like as a 6, and then f, just sec, just another option that maybe happen or that’s like that thing has this different function too?	143 guessing
	Interviewer	
144	Ok. Cool. Uhmmm, What is a function?	
	Tyler	
145	A function is kind of like a command or something, like when this happens you do this or make this move or something, like, do this, except, say, for example, it’s giving me, give like, the “a” button, key, a function for like maybe making a bright light, and so that every time you click that makes like a bright like, every time you click it, and then depends on the what the function reading is, depending of what type of flash there is or something?	145 relating
	Interviewer	
146	Ok. What is a variable?	

	Tyler	
147	A variable is things that, like there is multiple options or something, or multiple of some things, maybe multiple of oranges or apples.	147 relating
	Interviewer	
148	Alright. And how do functions and variables work together?	
	Tyler	
149	Like you give something like a function, but if you do that, like say the "a" thing, like it make a flash but maybe like the variable makes it so like, when you click it you don't just see like the same flash over again, so like maybe a different flash that comes up. like maybe switch off or randomly switch.	149 relating
	Interviewer	
150	Alright. And that my friend, is it.	

Coding
2 Relating to math
2 Understanding Racket
2 Giving example
4 Explaining value
8 Fumbling answer
12 Relating to math
14 Relating to math.
14 Expanding answer to difference between math and Racket
16 Relating to math
18 Moving beyond questioning to explain point
20 Clarifying
29 Not understanding
33 Not understanding
35 Relating to experienced math
35 Realizing similarities
38 Showing math understanding
43 Fumbling answer
49 Exploring the concept of function
56 clarifying
76 Fumbling
79 clarifying
81 guessing
83 guessing
90 recognizing
100 fumbling

104 hesitating
107 clarifying
113 fumbling
114 Pleasing interviewer
121 Realizing,
121 Asking for help
125 realizing
128 fumbling
132 clarifying
136 understanding
141 guessing
143 guessing
145 relating
147 relating
149 relating

Not understanding
8 Fumbling answer
43 Fumbling answer
76 Fumbling
100 fumbling
113 fumbling
128 fumbling
81 guessing
83 guessing
141 guessing
143 guessing
56 clarifying
79 clarifying
107 clarifying
132 clarifying
29 Not understanding
33 Not understanding
104 hesitating
121 Asking for help
114 Pleasing interviewer

at level
2 Giving example
4 Explaining value
38 Showing math understanding
145 relating
147 relating
149 relating

cognitive bridge
2 Relating to math
12 Relating to math
14 Relating to math.
14 Expanding answer to difference between math and Racket
16 Relating to math
35 Relating to experienced math
35 Realizing similarities
90 recognizing
121 Realizing,
125 realizing
136 understanding
18 Moving beyond questioning to explain point
49 Exploring the concept of function

Category	**Not understanding**	
Statement	**Property**	**Hypothesis**
Function?. That like it has multiple options of what it could be, so like, it could have multiple different functions, depending on what you give it.	8 Fumbling answer	Wanting to sound like he understands, but not quite getting it.
define is like stating something and a function is like giving it an option of sort.	43 Fumbling answer	
Like, this f would maybe equal function and then that would be like f timesing equals whatever x is plus 5	76 Fumbling	
That, this is the function that takes x, but this function takes that one, to that one?	100 fumbling	
(+ (f(x) 5))	113 fumbling	
Well, uhm, there would be… And it would be showing you what that function option…	128 fumbling	
Number?	81 guessing	

value?	83 guessing	
It's like, that, be like, takes like x, so that would be like saying another one so there would be 6 so there would be 12 no 11 and then	141 guessing	
Oh, that's giving y function, like as a 6, and then f, just sec, just another option that maybe happen or that's like that thing has this different function too?	143 guessing	
Like in coding or?	56 clarifying	Listening, and trying to understand.
One what?	79 clarifying	
Ok, I don't know how to …, I don't know how I would like write the equal sign.	107 clarifying	
Like not counting this part.	132 clarifying	
shrug	*29 Not understanding*	
What does the f stand for? Or is f just a letter?	33 Not understanding	Just not seeing.
I think so.	104 hesitating	
It's like saying these are like uh that is the function that's like the variable of the function, but then it's telling you the main problem down here and it's like that the variable option plus like the function or something?	121 Asking for help	
Is that right?	114 Pleasing interviewer	

Category	Cognitive Bridge	
Statement	Property	Hypothesis
It's saying times 3 of a value of something, I think. And it's defining what it is. And then it's kind of like a math problem, I guess. Like times 3 value, like maybe you have the value is an orange. You times that 3 times, and you have like maybe three oranges. Or depending on how many values there are. You have so many.	2 Relating to math	Tyler is beginning to understand that the concepts taught in Bootstrap are similar to the math he has learned.
You are timesing 3 by 1, so it's like 3 times 1.	12 Relating to math	
Then it just be like 1 … 5, be like 3 times 5, but the computer reads it differently. Like addition and subtraction or that thing in first, and then put the other two numbers with the space in between them.	14 Expanding answer to difference between math and Racket	
If or, like that.. or so it reads it that way. But we'd read a problem like that 3 times 5.	16 Relating to math	
Uhm in math… Oh, the problem is like you ah. When I normally see this, it's normally like there is like a letter here, and then normally like a	35 Relating to experienced math 35 Realizing similarities	

letter, and a sign and another number, then you times this by those, and then you get that, equals like this, and then you sort of divide from each side like …		
Kind of the same in a sense. And it like, it was like defining our function, and then that's like a random number and that's a random number, but you're doing the function like timesing this but when you get that you are timesing it by that.	90 recognizing	Having been introduced to algebra functional notation, Tyler can see the similarities.
t's like saying these are like uh that is the function that's like the variable of the function, but then it's telling you the main problem down here and it's like that the variable option plus like the function or something?	121 Realizing,	Tyler is gaining an understanding of functions.
No, they are the same thing, just in different form?	125 realizing	
11	136 understanding	
Three times-ing it by 0 and it. Each time the value changes you get a different answer. And sometimes the answer is big. Sometimes it is small.	18 Moving beyond questioning to explain point	Tyler has a good grasp of simple functions in Racket notation.
you were to finding like a key but when you gave that key a	49 Exploring the concept of function	Using Racket to grasp the concept of a function.

function of when you hit it like something happened like maybe you went up or depending upon what button or something.		

Was able to grasp algebraic functions, but not to the step of understanding composition of functions.

Vista Heights Middle School

Interviewee Information
Name: David
Age: 13
Grade: 7
Current Math: Algebra I, Math Tutor

Interview Context
Location: Vista Heights Middle School
Date: May 18, 2011
Time: 4:15 p.m.
Interview time: 12 minutes

	On board is written	
	(define (times3 value)	
	(* 3 value))	
	Interviewer:	
1	So what does this mean.	
	David:	
2	Whatever you put in value it would multiply that by 3.	2 answering
	Interviewer:	
3	So, if we go ahead and did	
	Write on board	
4	(times3 1)	
5	That means	

	David:	
6	3	6 answering
	Interviewer:	
7	3. And if we did a 5	
	Write on board	
8	(times3 5)	
	David:	
9	15	9 answering
	Interviewer:	
10	And 0	
	Write on board	
11	(times3 0)	
	David:	
12	0	
	Interviewer:	
13	Right. And a negative 2	
14	And 0	
	Write on board	
15	(times3 -2)	
	David:	
16	Negative 6	
	Interviewer:	
17	Ok. No problems there.	
18	Alright. What is value?	
	David:	
19	A variable, Value could be anything. In this case a number.	19 explaining
	Interviewer:	
20	How does value relate to a function?	

	David:	
21	Value could be like, if were talking like about a math could be x or y, a variable, like I said earlier. So it could be anything. Like value could be a string. But in this function it wouldn't work.	21 Explaining 21 exploring
	Interviewer:	
23	What happens with value in a function?	
	David:	
24	Are you talking about a computer function or math?	24 clarifying
	Interviewer:	
25	Take your pick.	
	David:	
26	For a computer function, whatever you did times3, whatever you put for that. Value would be whatever you put there, since it is a variable.	26 Explaining
27	But in math, it would just, same thing, variable, it could be anything.	27 realizing
	Interviewer:	
28	What if I said,	
	Write on board	
29	times3(value)=3value	
30	What does that mean?	
	David:	
31	That means that … times3 is the 3because it's the same thing, multiplying by value.	31 associating
	Interviewer:	
32	What do you mean "same thing"?	
	David:	
33	Because, 3 value, times 3 value, so they have to equal the same thing. So times3 would equal 3. So then, but value you could figure because there is nothing to define it. like	33 exploring
	Interviewer:	
34	Ok. What is the difference between	
35	this and this?	
	Referring to times3(value) and time3 function	

	David:	
36	One is a computer function, and one is a notation in math. In this case, it's saying that whatever you put in for value will be multiplied by 3. It's pretty much the same thing in the other case, it's just in a math format.	36 realizing
	Interviewer:	
37	Now, what if I wrote it as	
	Write on board	
38	f(x)=3x	
39	like that.	
	David:	
40	It's same thing. Because, functions are sort of like a variable also. And then x and x are the same thing. So function times x equals 3x, so function has to equal 3.	40 Showing new knowledge
	Interviewer:	
41	But what do you call this?	
	David:	
42	x is a variable.	
	Interviewer:	
43	Ok.	
	David:	
44	But in this case in equal value, if you match it up to the other equation.	44 exploring
	Interviewer:	
45	Right. So. Is there a difference between that and that, x and value?	
	David:	
46	No.	46 understanding
	Interviewer:	
47	Ok. My mistake.	
48	What is the different between x and value.	
	David:	
49	It is just written differently.	49 understanding
	Interviewer:	
50	Alright. Are there other ways you can represent this function? *Referring to f(x)=3x*	
	David:	
51	Yes, There can be different order for like x times f, or f times 3, or whatever. Or you could use different things for variable. Like, instead of x you could use y or a or whatever.	51 Demonstrating knowledge

	Interviewer:	
52	Can you draw this function as a graph?	
	David:	
	Draws straight line	*52 Answering wrong*
53	By the looks of it there wouldn't be a slope, let me see.	
54	Since because it's saying that equals that. But then multiply it so	
55	Depends on how you look at it, you could either think of it as a straight line, or think about it timeing 3 for the y , so then this one would be 3, and right there 6.	
	Interviewer:	
56	Now. What if I write something like, let's do this.	
	Write on board	
57	f(x) = x + 5	
	g(y) = 10 – y	
	f(g(x))	
58	What would f of g of x mean?	
	David:	
59	Are you trying to figure out the value for?	59 clarifying
	Interviewer:	
60	Just what it means.	
	David:	
61	fgx mean you are multiplying f time g times x, I could figure it out to see what it is.	61 fumbling
	Interviewer:	
62	Well, that's ok.	
63	So we were able to create a function here in math that matched a function in bootstrap. Could you write these here (*f(x) and g(y))* in Bootstrap?	
	David:	
64	Yes.	
	Interviewer:	
65	Would you do that?	
	David:	
66	Yes.	
	Interviewer:	
67	I'm not supposed to ask yes or no questions. Go ahead and write this in Bootstrap.	

	David:	
68	Do you want each of them separately, or make it so they equal each other?	68 clarifying
	Interviewer:	
69	Just ahhh, just what you see.	
	David:	
70	Do you want me to do circle of evaluations?	
	Interviewer:	
71	Circle of evaluations is great.	
	David:	
72	There is the first one.	72 trying
	…	
73	There is the second one.	
74	Might as well do the third one.	
	Interviewer:	
75	This has been very helpful for me. I appreciate it.	
76	I'll let you get back.	

C.2.1.2 Coding

2 answering
6 answering
9 answering
19 explaining
21 Explaining
21 exploring
24 clarifying
26 Explaining
27 realizing
31 associating
33 exploring
36 realizing
40 Showing new knowledge
44 exploring
46 understanding
49 understanding
51 Demonstrating knowledge
52 Answering wrong
59 clarifying
61 fumbling
68 clarifying
72 trying

Not Understanding
24 clarifying
52 Answering wrong
59 clarifying
61 fumbling
68 clarifying
72 trying

At Level
2 answering
6 answering
9 answering
19 explaining
21 Explaining
26 Explaining

Cognitive Bridge
21 exploring
27 realizing
31 associating
33 exploring
36 realizing
40 Showing new knowledge
44 exploring
46 understanding
49 understanding
51 Demonstrating knowledge

Category	**Not understanding**	
Statement	Property	Hypothesis
Are you talking about a computer function or math?	24 clarifying	Not sure of difference yet
Draws straight line	*52 Answering wrong*	*Wrong graph, probably rushed*
Are you trying to figure out the value for?	59 clarifying	
fgx mean you are multiplying f time g times x, I could figure it out to see what it is.	61 fumbling	Clearly not understanding syntax of f(g(x))
Do you want each of them separately, or make it so they equal each other?	68 clarifying	
There is the first one.	72 trying	

Category	**At Level**	
Statement	Property	Hypothesis
Whatever you put in value it would multiply that by 3.	2 answering	
3	6 answering	
15	9 answering	
A variable, Value could be anything. In this case a number.	19 explaining	
Value could be like, if were talking like about a math could be x or y, a variable, like I said earlier. So it could be anything. Like value could be a string. But in this function it wouldn't work.	21 Explaining	
For a computer function, whatever you did times3, whatever you put for that. Value would be whatever you put there, since it is a variable.	26 Explaining	

Category	**Cognitive Bridge**	
Statement	Property	Hypothesis
Value could be like, if were talking like about a math could be x or y, a variable, like I said earlier. So it could be anything. Like value could be a string. But in this function it wouldn't work.	21 exploring	David is looking for understanding – recognizing his knowledge is a bit short in this area
But in math, it would just, same thing, variable, it could be anything.	27 realizing	First answers hesitantly, but then solid – showing new understanding
One is a computer function, and one is a notation in math. In this case, it's saying that whatever you put in for value will be multiplied by	36 realizing	

3. It's pretty much the same thing in the other case, it's just in a math format.		
That means that … times3 is the 3because it's the same thing, multiplying by value.	31 associating	Using his background to explain
Because, 3 value, times 3 value, so they have to equal the same thing. So times3 would equal 3. So then, but value you could figure because there is nothing to define it. like	33 exploring	
But in this case in equal value, if you match it up to the other equation.	44 exploring	
No.	46 understanding	
It is just written differently.	49 understanding	Shows clear understanding
Yes, There can be different order for like x times f, or f times 3, or whatever. Or you could use different things for variable. Like, instead of x you could use y or a or whatever.	51 Demonstrating knowledge	Understanding has come to the point that he can elaborate
It's same thing. Because, functions are sort of like a variable also. And then x and x are the same thing. So function times x equals 3x, so function has to equal 3.	40 Showing new knowledge	

Lehi High School

Interviewee Information
Name: Sally
Age: 17
Grade: 12
Highest Math: Personal Finance

Interview Context
Location: Lehi High School
Date: November 7, 2011
Time: 12:30 p.m.
Interview time: 7 minutes

Interviewer:
So first of all, what did you think of this program.
Sally:
I liked it. I learned more about math than I do in my regular math class.
Interviewer:
How's that?
Sally:
It's like, in here it is harder, but my regular math class really isn't it for me.
Interviewer:
Uh huh
Sally:
So, I like it a lot. Like, learning more about math and everything.
Interviewer:
OK. Ok. Uh. So for this class, suppose I say

```
(define (f x))
        (+ 3 x))
```

and then I said

```
(f 0)
```

what do I get back
Sally:
You get 3
Interviewer:
And if I said (f -2)
Sally:
Get 1
Interviewer:
Ya. Ok. Now, if I said something like
$f(x)=x+3$
What is the difference between that and that.
Sally:
I… don't kno… so like the x is over there, and it's in its own parentheses

Interviewer:
Uh huh
Sally:
And this is written different than that is.
x+3 … different …
Interviewer:
Have you ever seen this like this before?
Sally:
Ya!
Interviewer:
OK, where did you see that at?
Sally:
In my math classes before, and I've only seen that here.
Interviewer:
Right. OK, so this one is called Racket, I don't know if they told you the name of the language – Bootstrap or whatever you want to call it
Sally:
Ya
Interviewer:
OK, this one is algebra.
Do they mean the same thing?
Sally:
Ya!
Interviewer:
OK. Just written differently. Grreat…
OK, so now, how did you do with the Pythagorean Theorem type stuff.? and those things?
Sally:
I have no idea
Interviewer:
Alright. Suppose I wrote

```
(define (f x)
        (+ 3 x))
```

and then I said

```
(define (g y)
        (- g 2))
```

Now what if I said

```
(f (g 6))
```

Sally:
So first I would do 6 – 2, that would be four
Then I would add it to 3 and that would be 7

Interviewer:
OK, there you go.
But what if I wrote it this way?

f(x)=3+x
g(y)=g-2

and then said f(g(y))
Sally:
Then it would be like the same thing. I think
Interviewer:
Right…
Sally:
So that y is a …
Interviewer:
So is this scary?
Sally:
Ya! (*nervous laugh*)
Interviewer:
Why is that scary?
Sally:
It just looks like it is written in a different format, a lot different than you write in.
Interviewer:
Uh huh
Sally:
Confusing
Interviewer:
Confusing…
I can go with that.
So did that not make sense on the test when you had to do something like that.
Sally:
No! I had no idea what I was doing on that part.
Interviewer:
OK
Sally:
It was confusing.
Interviewer:
Let's do, how about conditionals… Did you get conditionals?
Sally:
Conditionals are what?
Interviewer:
So let's see here
```
(define (f x)
        [(cond (x<2
```

Sally:
Oh yeah, we explained these.
Interviewer:
    [(x<2) (+ x 3)]
    [(x>= 2) (+ x 5)]

Does that make sense?
Sally:
Ya.
Interviewer:
but if you come to this
f(x) = x<2 then x+5 and the x>=2 x+5.. oh 3
Sally:
It's more confusing written that way.
Interviewer:
definitely more confusing
Sally:
Ya
Interviewer:
OK, that's really
If you were back at a math class, and you started working with this, would you feel more comfortable now?
Sally:
N…o, like I could like do it, but I think it would be hard because I learned like that. But I want to do it a different way.
Interviewer:
OK. That's great. Thank you very much.

Interviewee Information
Name: Ned
Age: 17
Grade: 12
Current Math: Algebra I

Interview Context
Location: Lehi High School
Date: November 11, 2011
Time: 12:50 p.m.
Interview time: 7 minutes

Interviewer:
If I say
define some function f, and it has one parameter – and it is x+3

Ned:
32, 37 , that or that– looks good I guess
Interviewer:
You know what this means
Ned:
Ya
Interviewer:
What does it mean?
Ned:
You have to define what x is
Interviewer:
Ya, so what if I said, if I called
(f 5)
Ned:
You would have to figure out what 5 is
Interviewer:
huh?
Ned:
not 5, f, right?
Interviewer:
OK, so if I put this in, then later on I put this in, what are you going to get back?
Ned:
You are going to have to figure out the f
Interviewer:
No, this is the name of the function, and this is the variable, and then here is the body of the function.
Ned:
So I have to put up the 5, so 5 and 3
Interviewer:
So what do you get back?
Ned:
15
Interviewer:
It's plus
Ned:
oh, 8
Interviewer:
ok, so now (f 2)
Ned:
3 and 2 is 6, gotcha
Interviewer:
and (f -1)
Ned:
negative 2, 3?

Interviewer:
negative 1 plus 3, that'd be 2
Ned:
ya 2
Interviewer:
No problem. OK, so now
(define (g y)
that's a y
Ned:
awesome
Interviewer:
and call it minus x and 2
so if I said y of 2
Ned:
That'd be 2, 2 times …
Interviewer:
yea…
let's try an easier one. (y 6)
Ned:
4
Interviewer:
4, so
yea, no big deal.
Let's say .. another function …
(g (h z))
So if I said h of 5
Ned:
So it'd be 5 and 2
Interviewer:
5 plus 2, right. No wait a second
Ned:
Then you close that, right?
5 and 2 is 7 and plus right 3 there
Interviewer:
OK, so you're saying you want to do the
Ned:
5 and 2
Interviewer:
OK, so you've got 5 and 2, why do you want the 2. Don't you want to come to g first?
Ned:
yea. Great.
Interviewer:
Or do you want to come to f first?
Ned:
Oh yeah, cause f is closer.

Interviewer:
yeah, f is closest
You are calling f of z
Ned:
you have z
Interviewer:
which is going to be
Ned:
3, no. 5, no.
Interviewer:
And so you've got
(+ 5 3)
Ned:
z, yeah, so + 5 3, yeah.
Interviewer:
Is?
Ned:
8, cause 5 and 3 is 8
Interviewer:
And then you'd take that 8 and you put it, calling g of this, and you are passing in that 8
Ned:
yeah. So you would put, z, so wouldn't you put like an f , find …
So I'm thinking you would put the 5 and the 2 , not 2, 3 – then 5 and 3 would be 8
Interviewer:
right
Ned:
So you minus 2, right?
Interviewer:
Uh huh, so it'd be 4?
Ned:
6
Interviewer:
6, yeah. There you go. Alright
Now tell me, in your algebra class, have you seen anything like this
f(x)=x+3
Ned:
No
f over x…
Interviewer:
Yeah, something like that
So on the exam, no it wasn't an exam – evaluation, you didn't answer because you didn't know what this was
Ned:
yeah

Interviewer:
How about the conditionals. DO your remember how those worked?
```
(define (j var)
        (cond
                ([var <  2] (…)
                ([var >= 2] (then do something else)
```
Does that make sense
Ned:
yes. So if var is =2 or not equal to then …
Interviewer:
So if I went ahead and said 6
Ned:
so I'd be 5 and 2, no 6 and 2
Interviewer:
So you got var >= 2 so you come back as
Ned:
5
Interviewer:
Cool
Ned:
Cool
Interviewer:
Thank you very much.

Interviewee Information
Name: Dan
Age: 18
Grade: 12
Current Math: Algebra I, 2 years ago

Interview Context
Location: Lehi High School
Date: November 11, 2011
Time: 1 p.m.
Interview time: 9 minutes

Interviewer:
So for this class, you've learned how to do something like
```
(define (f x)
        (+ x 2))
```
Dan:
Yeah.
Interviewer:

So later on if I call (f 5) what do I get?
Dan:
uhm, you get 7
Interviewer:
7. right.
And if I called (f -1)
Dan:
then 1
Interviewer:
1, right. OK. No big deal there.
So now, what if I did another function and I said
(define (g y)
            (- x 3))
oops x, You can't do that
So if I said (g of 6)
Dan:
You'd get 3
Interviewer:
(g 1)
Dan:
That'd be 4? Oh, that's minus. It'd be 2.
Interviewer:
negative 2. Cool. Alright – now.
Let's see, if you went to something like
(define h z)
and I say, and I want to call g of f of z *(g (f z)))*
So if I did
(h 6)
Dan:
so is this minus six
Interviewer:
so the function name is h
Dan:
Oh, h=6
Interviewer:
you call h with a 6
Dan:
z is 6, and then f is plus 2 or times 2?
Interviewer:
It's a plus 2
Dan:
OK, plus 2, so that'd be 6 h
Interviewer:
uh huh!

Dan:
And then that'd be 5
Interviewer:
right. cool.
So now, in your math class earlier did you see something like
f(x)=x + 2
Dan:
Yeah
Interviewer:
You've seen that before?
What does it mean?
Dan:
f is x, like that's the name
Interviewer:
uh huh
Dan:
f is equal to x+2 which means it's just 2
Interviewer:
OK, so if I called, or did f(0), then what would I get?
Dan:
2
Interviewer:
2. And if I said f(1)
Dan:
3
Interviewer:
3. And now, what is the difference between these two things here?
Dan:
Uhm, the first one is adding something to it before you ever do a function
Interviewer:
uh huh
Dan:
And the second one is you just hold it and let the ??? take place of it
Interviewer:
OK. Does it do kinda the same thing though?
Dan:
Uhm. Yes and no. It doesn't come out with the same answer but it does basically the same thing.
Interviewer:
OK. Why doesn't it come out with the same answer?
Dan:
Because the equal is there with the plus 2

Interviewer:
OK. What if I said something like, uhhhh I'm going to do
g(y)=y-3
Does that make any sense?
Dan:
Yeah. Negative 6 right there.
Interviewer:
If I did a, well if I called f of g with a 6
Dan:
So it'd be 0? or
Interviewer:
calling g with a 6 and then you'd get back
Dan:
Uh, oh wait. Why are we talking about this?
Interviewer:
Why are we talking about this?
Dan:
No. y are we talking about
Interviewer:
Oh. This y right here
Dan:
Oh, I thought we were talking about the y over there.
Interviewer:
No. So, let's go back away from this.
Dan:
So that'd be just 3
Interviewer:
Yep, that'd be just 3. Now. What if we did a function called h and z
Ignore all that over there. Oh, I'm sorry, let's do this. what if I said
z(f(7))
Dan:
So, close to 9, g of g 6
Interviewer:
OK. Pretty close. So, is this doing anything different? These two things here?
Dan:
What things?
Interviewer:
This and this?
Dan:
No really
Interviewer:
Yeah. Kinda the same thing, aren't they
Dan:
Yeah. Just different answers

Interviewer:
OK… So let's go, uhm…. Let's say define a function called a, it takes parameter called x. And it has a cond. So we'll say a < 2 then return a
oh. I'm backwards, aren't I
less than a 2 , the I would say + a 3
Then if I said >= a 2 then I'd return 6
Dan:
Uhm. If you have less that 2, then you have plus 3
and if it's over.
Interviewer:
So if I said (a 1)
Dan:
So negative 3 right there.
And then that'd be 4 , uhm… 6, 6, 12
Interviewer:
OK. IS that how this kinda works?
Dan:
Yeah.
Is that it.
Interviewer:
That's it.

Interviewee Information
Name: Mike
Age: 15
Grade:
Current Math: Algebra I

Interview Context
Location: Lehi High School
Date: November 11, 2011
Time: 1:10 p.m.
Interview time: 5 minutes

Interviewer:
What if I create on function, call f, give it one parameter, call it x, and then say + x 1
and then later on I say f of 6
Mike:
It was one of … that they never taught me, so I don't know.
Interviewer:
So if you call this function, passing a 1, …
So if you put a 6 right here, then what happens?

Mike:
Uhm…
Interviewer:
Then it just becomes?
It would be + 6 and 1
Mike:
7?
Interviewer:
Good! That is all it is. And if I said f of 10?
Mike:
So you just do 10 + 1 which is 11.
Interviewer:
Yep.   Now if I did another one, g and y.
And I said – y and 1
And if I said g of 6
Mike:
confused
Interviewer:
g and 6
Mike:
So 6 -1 one would be 5
Interviewer:
Yeah.
And now what if I said define h and z and said g of f of 3
Mike:
long pause
times 2 3?
Interviewer:
K, so first of all think about this one right here.
Think about that portion right here, what happens?
If I just wrote it out this way, f of 3, what would you get?
Back up here.
Mike:
Oh, OK. 5 2 minus
Interviewer:
Yeah, you get 4. Now you've got a 4.
And so now if I said g of 4, what would I get?
Mike:
5?
Interviewer:
g, that's a minus.
Mike:
Oh, so… What?
Interviewer:
OK, so if I said g of 4

Mike:
Oh, minus, yeah. Negative
Interviewer:
4 minus 1
Mike:
4 minus 1 so 3?
Interviewer:
OK.
Mike:
I was just thinking you don’t want …
Interviewer:
Right. So you take all this.
g of 4, so it comes back as 3.
Mike:
Yeah.
Interviewer:
So have you seen this kind of notation in your algebra class
show piecewise function
Mike:
Not yet
Interviewer:
OK.
Alright, that is all I need.
Thanks.

APPENDIX F CONSENT FORMS

Bootstrap Participation Parental Consent Form

Investigating the effects of Computer Programming on Mathematical Understanding

Parent Permission Consent Form

(i) **Introduction**
Your child's school is conducting a project to examine the effect studying core programming principles might have on students' mathematical problem solving abilities. Your child was selected to represent your school in this effort or because your child has voluntarily enrolled in the Communications Technology course. This research study is being conducted by Dr. Peter Rich, from the McKay School of Education, and Dr. Geoffrey Wright from the college of Engineering, at Brigham Young University.

(ii) **Procedures**
This research consists of three parts: establishing a baseline, teaching programming, and measuring growth.
Establishing a baseline: At the beginning of the course, your child will be asked to complete a 30-minute mathematics problem-solving exam. This exam has no bearing on course grades or assignments.
Teaching programming: Students that have enrolled in Communications Technologies will be taught several core programming concepts, implementing this by creating several small programs and games.
Measuring growth: After the end of the semester, your child will be asked to take another mathematics problem-solving exam. You may also be interviewed by researchers to talk about your experience. These measures will have no bearing on your course grade.

(iii) **Risks/Discomforts**
Although risk is minimal from participating in this study, you may feel discomfort by being asked to complete the pre and post-test problem solving exams.

(iv) **Benefits**
Benefits of participating in this study are:
1. Greater understanding of how computer programming and mathematics relate.
2. Increased understanding of computer programming (for Communications students).
3. Several electronic products (e.g., games) that you will create for your own and others' use (for Communications students).

(v) **Confidentiality**
Researchers will keep your personal data in a secured, locked filing cabinet. Electronic data will be password-protected and shared only between researchers. Any published data will remain anonymous to the public. Pseudonyms will be used when referring to specific students.

(vi) Compensation
There is no compensation for participation.

(vii) **Participation**
Participation in this research study is voluntary. You and your child have the right to withdraw at anytime or refuse to participate entirely without jeopardy to your class status, grade or standing within the school.

(viii) **Questions about the Research**
If you have questions regarding this study, you may contact P. Rich, PhD, at 801-422-1171, peter_rich@byu.edu or G. Wright, PhD, at 422-7804, ge.wright@gmail.com.

(ix) **Questions about your Rights as Research Participants**
If you have questions regarding your rights as a research participant, you may contact the BYU IRB Administrator at (801) 422-1461, A-285 ASB, Brigham Young University, Provo, UT 84602, irb@byu.edu.

I have read, understood, and received a copy of the above consent and desire of my own free will to participate in this study.

Name (please print): ______________________________________Date:

Signature:

Parent Name (please print): ______________________________________Date:

Parent Signature:

Bootstrap Participation Student Form

What is this study about?
My name Geoffrey Wright. I am from Brigham Young University. I would like to invite you to take part in a research study. Your parent(s) know we are talking with you about the study. This form will tell you about the study to help you decide whether or not you want to be in it.

In this study, we want to learn about *computer programming and making video games. We are studying whether learning computer programming will help you better learn math.*

What am I being asked to do?
If you decide to be in the study, we will ask you to *learn how to design and make computer video games. Take a short math quiz before and after learning to make the video games, and be interviewed by a BYU student about your experience programming and design the video games.*

What are the benefits to me for taking part in the study?
There are not any specific benefits to taking part in the study – besides, obviously learning how to design and make video games.

Can anything bad happen if I am in this study?
There is nothing bad that will happen by participating in this study. Your scores from the math quiz won't be shared with any of your teachers, friends, or parents. The course will be fun for you – who doesn't want to learn how to be a video game programmer?!

Who will know that I am in the study?
We won't tell anybody that you are in this study and everything you tell us and do will be private. Your parent may know that you took part in the study, but we won't tell them anything you said or did, either. When we tell other people or write articles about what we learned in the study, we won't include your name or that of anyone else who took part in the study.

Do I have to be in the study?
No, you don't. The choice is up to you. No one will get angry or upset if you don't want to do this. And you can change your mind anytime if you decide you don't want to be in the study anymore.

What if I have questions?
If you have questions at any time, you can ask us and you can talk to your parent about the study. We will give you a copy of this form to keep. If you want to ask us questions about the study, call or email

Dr. Geoffrey A. Wright
801-422-7804 230 G Snell Building, Brigham Young University, Provo, Utah 84664

Do you have any questions about the study now?

****

IF YOU WANT TO BE IN THE STUDY, SIGN AND PRINT YOUR NAME ON THE LINE BELOW:

____________________________________ _________________
Sign your name Date

Bootstrap Control Group Parent Consent Form

Investigating the effects of Computer Programming on Mathematical Understanding

Parent/Guardian Permission Consent Form

(i) Introduction
Your child's school is conducting a project to examine the effect studying core programming principles might have on students' mathematical problem solving abilities. Your child was selected to represent your school in this effort. This research study is being conducted by Dr. Peter Rich, from the McKay School of Education, Dr. Geoffrey Wright and Robert Lee from the college of Engineering, and Dr. Keith Leatham and Kiya Heaton from the college of Physical and Mathematical Sciences at Brigham Young University.

(ii) Procedures
This research consists of three parts: establishing a baseline, and measuring growth.
Establishing a baseline: At the beginning of the study your student will be asked to complete a 30-minute mathematics problem-solving exam. This exam has no bearing on course grades or assignments.
Measuring growth: After the end of the study you child will be asked to take another mathematics problem-solving exam. These measures will have no bearing on your course grade.

(iii) Risks/Discomforts
Although risk is minimal from participating in this study, you may feel discomfort by being asked to complete the pre and post-test problem solving exams.

(iv) Benefits
There are no direct benefits to participating in this study.

(v) Confidentiality
Researchers will keep your personal data in a secured, locked filing cabinet. Electronic data will be password-protected and shared only between researchers. Any published data will remain anonymous to the public. Pseudonyms will be used when referring to specific students.

(vi) Compensation
There is no compensation for participation.

(vii) **Participation**
Participation in this research study is voluntary. You or your child have the right to withdraw at anytime or refuse to participate entirely without jeopardy to your class status, grade or standing within the school.

(viii) **Questions about the Research**
If you have questions regarding this study, you may contact P. Rich, PhD, at 801-422-1171, peter_rich@byu.edu or G. Wright, PhD, at 422-7804, ge.wright@gmail.com.

(ix) **Questions about your Rights as Research Participants**
If you have questions regarding your rights as a research participant, you may contact the BYU IRB Administrator at (801) 422-1461, A-285 ASB, Brigham Young University, Provo, UT 84602, irb@byu.edu.

I have read, understood, and received a copy of the above consent and desire of my own free will to participate in this study.

Name (please print): ______________________________________Date:

Signature:

Parent Name (please print): ______________________________________Date:

Parent Signature:

Bootstrap Control Group Student Form

What is this study about?
My name Geoffrey Wright. I am from Brigham Young University. I would like to invite you to take part in a research study. Your parent(s) know we are talking with you about the study. This form will tell you about the study to help you decide whether or not you want to be in it.

In this study, we want to learn about *computer programming and making video games. We are studying whether learning computer programming will help you better learn math.*

What am I being asked to do?
If you decide to be in the study, we will ask you simply take two short math quizzes during your math class.

What are the benefits to me for taking part in the study?
There are not any specific benefits to taking part in the study.

Can anything bad happen if I am in this study?
There is nothing bad that will happen by participating in this study. Your scores from the math quiz won't be shared with any of your teachers, friends, or parents.

Who will know that I am in the study?
We won't tell anybody that you are in this study and everything you tell us and do will be private. Your parent may know that you took part in the study, but we won't tell them anything you said or did, either. When we tell other people or write articles about what we learned in the study, we won't include your name or that of anyone else who took part in the study.

Do I have to be in the study?
No, you don't. The choice is up to you. No one will get angry or upset if you don't want to do this. And you can change your mind anytime if you decide you don't want to be in the study anymore.

What if I have questions?
If you have questions at any time, you can ask us and you can talk to your parent about the study. We will give you a copy of this form to keep. If you want to ask us questions about the study, call or email

Dr. Geoffrey A. Wright
801-422-7804 230 G Snell Building, Brigham Young University, Provo, Utah 84664

Do you have any questions about the study now?

****

IF YOU WANT TO BE IN THE STUDY, SIGN AND PRINT YOUR NAME ON THE LINE BELOW:

____
Sign your name Date