The Effect of Problem-Solving Instruction on the Programming Self-efficacy and Achievement of Introductory Computer Science Students

by

Elizabeth Maddrey

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computing Technology in Education

Graduate School of Computer and Information Sciences
Nova Southeastern University
2011

المنارة للاستشارات

www.manaraa.com

UMI Number: 3466275

All rights reserved

INFORMATION TO ALL USERS
The quality of this reproduction is dependent on the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript
and there are missing pages, these will be noted. Also, if material had to be removed,
a note will indicate the deletion.

UMI
Dissertation Publishing

UMI 3466275
Copyright 2011 by ProQuest LLC.
All rights reserved. This edition of the work is protected against
unauthorized copying under Title 17, United States Code.

ProQuest

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346

المنارة للاستشارات

www.manaraa.com

We hereby certify that this dissertation, submitted by Elizabeth Maddrey, conforms to acceptable standards and is fully adequate in scope and quality to fulfill the dissertation requirements for the degree of Doctor of Philosophy.

Steven R. Terrell, Ed.D. Chairperson of Dissertation Committee	Date
Laurie Dringus, Ph.D. Dissertation Committee Member	Date
Martha Snyder, Ph.D. Dissertation Committee Member	Date

Graduate School of Computer and Information Sciences
Nova Southeastern University
2011

المنارة للاستشارات
www.manaraa.com

An Abstract of a Dissertation submitted to Nova Southeastern University as a Partial Fulfillment of the Requirements for the degree of Doctor of Philosophy in Computing Technology in Education

The Effect of Problem-solving Instruction on Programming Self-Efficacy and Achievement of Introductory Computer Science Students

by
Elizabeth Maddrey

June 29, 2011

Research in academia and industry continues to identify a decline in enrollment in computer science. One major component of this decline in enrollment is a shortage of female students. The primary reasons for the gender gap presented in the research include lack of computer experience prior to their first year in college, misconceptions about the field, negative cultural stereotypes, lack of female mentors and role models, subtle discriminations in the classroom, and lack of self-confidence (Pollock, McCoy, Carberry, Hundigopal, & You, 2004). Male students are also leaving the field due to misconceptions about the field, negative cultural stereotypes, and a lack of self-confidence. Analysis of first year attrition revealed that one of the major challenges faced by students of both genders is a lack of problem-solving skills (Beaubouef, Lucas & Howatt, 2001; Olsen, 2005; Paxton & Mumey, 2001).

The purpose of this study was to investigate whether specific, non-mathematical problem-solving instruction as part of introductory programming courses significantly increased computer programming self-efficacy and achievement of students.

The results of this study showed that students in the experimental group had significantly higher achievement than students in the control group. While this shows statistical significance, due to the effect size and disordinal nature of the data between groups, care has to be taken in its interpretation. The study did not show significantly higher programming self-efficacy among the experimental students. There was not enough data collected to statistically analyze the effect of the treatment on self-efficacy and achievement by gender. However, differences in means were observed between the gender groups, with females in the experimental group demonstrating a higher than average degree of self-efficacy when compared with males in the experimental group and both genders in the control group. These results suggest that the treatment from this study may provide a gender-based increase in self-efficacy and future research should focus on exploring this possibility.

www.manaraa.com

Acknowledgments

It is with the deepest gratitude that I thank all those who made this thesis possible. Foremost among them is Dr. Steven Terrell, my committee chair. Without his perseverance, encouragement, and belief in my abilities, I would not have completed this journey. I also sincerely appreciate the help of my committee members, Dr. Laurie Dringus and Dr. Martha Snyder. Their time and feedback throughout this process have been invaluable.

The programming faculty of Seminole State College of Florida also played an invaluable role in this study. Particular thanks go to Dick Grant for providing an entrée into the department as well as David Taylor, Jatin Shah, and Rebekah Gabel for working with me and their students to achieve the timely and successful completion of this study.

Finally, I would like to thank my husband, Tim Maddrey, for encouraging and believing in me through the many ups and downs of this process. I would also like to thank my dad, Dr. Dennis Perry, my mom, Linda Perry, M.Ed., and my sister Dr. Lynellen Perry, for placing such a priority on education and giving me the tools and desire to be a lifelong learner.

www.manaraa.com

Table of Contents

www.manaraa.com

المنارة للاستشارات
www.manaraa.com

List of Tables

المنارة للاستشارات

www.manaraa.com

Chapter 1

Introduction

Background

The U.S. Department of Labor Bureau of Labor Statistics' *Occupational Outlook Handbook 2008-2009 Edition* (2007) predicts that employment opportunities in computer related disciplines will be among the fastest growing occupations through 2016. This growth is projected to be "much faster than average" with an increase in employment opportunities of 37% or more. With this positive job outlook, it is reasonable to expect a rise in interest in a computing discipline as a college major. According to the most recent issue of the U.S. Department of Education's National Center for Education Statistics *Digest of Education Statistics* (2007), conferment of degrees in computing disciplines increased between the years 2005 and 2006. Despite this increase, the number of students enrolling in the discipline is not predicted to be adequate to meet the needs of industry.

Since the foundational article by Camp in 1997, the *shrinking pipeline* has become a term used to describe the steadily decreasing number of female students entering college with an intention to major in a computing discipline. It has since been identified as a continuing and well-known problem within the field (Berkelaar, Kisselburgh, & Buzzanell, 2008; Norris, Barry, Fenwick, Reid & Rountree, 2008; Powell 2008; Rafieymehr, 2008; Rieksts & Blank, 2008; Sloan & Troy, 2008; Van Sickle, 2008; Wilson, 2008). Because the rate of increase in enrollment for males is significantly higher than that of females, creating a gender balance in the field of computing is critical if the overall number of graduates is to increase. The number of female students entering a

www.manaraa.com

computing major is small and shrinks further after the first year. Some postulate the cause of dropout for both genders is a lack of achievement in initial classes, which pushes them to consider, and ultimately choose, other majors (Adya, 2008; Biggers, Brauer, & Yilmaz, 2008; Lewis, Smith, Belanger, & Harrington, 2008; Moskal, Lurie, & Cooper, 2004, Norris et al., 2008; Powell, 2008; Sloan & Troy, 2008; Wilson, 2008).

Moorman and Johnson (2003) note that the need for smart, capable, and creative people in the computing disciplines continues to grow and women provide insights and perspectives that their male peers miss. Rosser (2005) detailed the importance of including women in the technological workforce as technology becomes an increasing element of all aspects of modern society. Rosser further points out that a lack of participation in computer science by women gives rise to the omission of many features and products that are needed and desired by women. This, in turn, continues the decline in female participation as the technology becomes increasingly male oriented. In addition to these problems, Rosser notes that males are more likely to focus on the purely technical components of software systems during development; this omits consideration of a majority of "soft-systems" that focus on human factors and human computer interaction. Since males typically interact with systems in a more technically driven manner, the omission of soft-systems has a negative impact primarily on women. Berkelaar et al. (2008) point out that this lack of female participation deprives women of access to an entire field of stable, high-paying jobs. They further note that with the predicted growth of computer jobs as indicated by the Department of Labor and the shrinking enrollment of both genders in computers, women are needed in the field in

المنارة للاستشارات

www.manaraa.com

order to prevent a loss of national competitiveness as there will not be enough men to fill the gap.

Academics and professionals have been studying the gender gap in computing for over a decade using a variety of approaches, including single-gender classes and mentoring programs. The majority of the findings indicate that the gender gap problem begins with recruiting women into the field (Hart, Early, & Brylow, 2008; Hu, 2008; Owens & Matthews, 2008; Rafieymehr, 2008; Rieksts & Blank, 2008; Sands, Moukhine, & Blank, 2008; Van Sickle, 2008) and continues with a need to retain them once they begin a course of study (Biggers et al., 2008; Cohoon, Wu, & Luo, 2008; Edmondson, 2008; Klawe & Leveson, 1995; Lewis et al., 2008; Lopez, Schulte, & Giguette, 2005; Papastergiou, 2008; Scragg & Smith, 1998; Powell, 2008; Sloan & Troy, 2008; Wilson, 2008). The primary reasons for the gender gap presented in the research include lack of computer experience prior to their first year in college (Hu, 2008; Powell, 2008; Van Sickle, 2008), misconceptions about the field (Beaubouef & McDowell, 2008; Owens & Matthews, 2008; Rafieymehr, 2008, Sands et al., 2008; Sloan & Troy, 2008; Van Sickle, 2008), negative cultural stereotypes (Adya, 2008; Edmondson, 2008; Sands et al. 2008; Van Sickle, 2008), lack of female mentors and role models (Edmondson, 2008; Powell, 2008), subtle discriminations in the classroom (Edmondson, 2008; Sands et al., 2008), and lack of self-efficacy (Cohoon et al., 2008; Kumar, 2008; Norris et al., 2008; Pollock, McCoy, Carberry, Hundigopal, & You, 2004; Powell, 2008; Sloan & Troy, 2008). Lack of self-confidence, misconceptions about the field, and negative cultural stereotypes have also been shown to deter males from pursuing a computer science major (Beaubouef &

المنارة للاستشارات
www.manaraa.com

McDowell, 2008; Biggers et al., 2008; Joseph, 2008; Lewis et al., 2008; McInerney, DiDonato, Giagnacova, & O'Donnell, 2006).

Wilson (2006) confirmed that even in situations where no achievement or skill difference was found between male and female students, female students consistently had lower confidence in their ability with computers. Further, female computer science majors were found to have less confidence in their ability than male non-majors. Madigan, Goodfellow, and Stone (2007) found that students, regardless of gender, tended to perceive their computer skills as more developed than they actually were. However, they also found that female students continue to have considerably lower self-efficacy when it comes to completing tasks on a computer, causing female students to hesitate to enroll in computer science classes or take on a computer science major. Those females who do initially enroll are prone to dropping out due to a lack of belief in their ability to complete the degree successfully. Madigan et al. recommend that instructors in the computer science field find ways to build females' confidence incrementally through the course of their computer science studies but offer no suggestions as to what methods may be beneficial. Analysis of first year attrition revealed that one of the major challenges faced by students of both genders is a lack of problem-solving skills (Beaubouef, Lucas & Howatt, 2001; Jin, 2008; Kumar, 2008; Norris et al., 2008; Olsen, 2005; Paxton & Mumey, 2001; Pulimood & Wolz, 2008; Ragonis & Hazzan, 2008).

Problem Statement

The problem investigated was the high attrition of students in introductory programming courses, which are generally the first courses required in computer science

المنارة للاستشارات
www.manaraa.com

majors. Attrition from the programming courses typically leads to a change in major. This, in turn, contributes to the decline of available computer professionals despite job growth in this arena. Beckwith et al. (2006) note that first experiences in computing serve to form the foundation for computing self-efficacy and that early failures, or perceived failures, generally set the tone for poor self-efficacy in the future despite future successes. Other researchers note that female students appear to lose interest in technology in the later elementary years, suggesting that interventions should take place as early in the school experience as possible (Cady & Terrell, 2007; Hart, Early, & Brylow, 2008; Owens & Matthews, 2008; Rieksts & Blank, 2008).

The small quantity of research surrounding problem-solving has focused on mathematics as the vehicle for mastering these skills, which can bias students against computing based on a misperception of their math ability. Lemire (2002) highlights the lack of research supporting the widely accepted idea that problem-solving skills learned in one discipline, such as math, can be easily transferred to other disciplines without specific instruction. He provides examples demonstrating the lack of transferability of these skills and suggests that while some students may be able to make the transfer on their own, for skills to universally be applicable in a particular domain they will need to be taught in that domain. Ali (2005) notes that critical thinking and problem-solving skills are necessary in computer science and should become a more explicit part of the undergraduate computer science curriculum.

المنارة للاستشارات

www.manaraa.com

Relevance and Significance

This research study will add relevant insight to the current understanding of the causes and potential solutions of the decline in computer science enrollment by examining whether specific instruction in non-mathematical problem-solving skills significantly increased the programming self-efficacy and achievement of students in their first programming course. With a better understanding of what affects the self-efficacy of students in the computing sciences, teachers at all levels will be better able to encourage enrollment and persistence in computer science disciplines. Previous studies designed to examine the shrinking pipeline problem have focused primarily on increasing female enrollment by identifying and reducing negative stereotypes and creating mentorship relationships between young women who express an interest in computer science and women currently working in the field. These two approaches, while potentially beneficial, are more effective with young women who are already predisposed to remain in the field and do not address the decline in male enrollment at all (Cohoon et al., 2008; Hu, 2008; Pollock et al., 2004).

Other researchers have noted the link between problem-solving and computer science, however this research focused on computer science as a pedagogical tool for problem-solving instead of vice versa (Teague, 2002; De Palma, 2001; Colley, Henry, Holmes & James, 1996; Joiner, Messer, Littleton, & Light, 1996). Jin (2008) discussed the impact of problem-solving instruction on the achievement of college students, but did not investigate any relationships to self-efficacy or retention. Kumar (2008) investigated the effect of repeated drill sessions, an alternative definition of the term problem-solving

المنارة للاستشارات

www.manaraa.com

than the one used in this study, on the self-confidence of college-level females. Pulimood and Wolz (2008) used the same definition of problem-solving as Kumar and investigated the effect of working in groups on the retention of college level females. The results from the present study will facilitate understanding of the proper role of problem-solving as a discipline in computer science education.

Broader impacts that will be realized from this study include adding to the literature surrounding the decreasing enrollment problem in computer science. The shrinking pipeline problem is of significant concern as the demand for qualified computer science practitioners continues to grow. Identification of possible solutions to encourage entry and retention of both genders into computing disciplines is a beneficial addition to the field. Conclusions derived from this study may provide insight into retention and successful graduation and employment of both genders.

Barriers and Issues

It is believed that the reason research on the interaction of problem-solving and self-efficacy has not been previously undertaken is due to the common assumption that the math courses required as prerequisites to computer science education are sufficient introduction to the analytical thinking skills necessary for success in computer science (Joshi & Schmidt, 2006; Lemire, 2002; Colley et al., 1996).

Research Hypotheses

The following hypotheses were investigated as part of this study:

Hypothesis One: Students who receive instruction in non-mathematical problem-solving and critical thinking skills prior to programming instruction will exhibit

المنارة للاستشارات

www.manaraa.com

significantly higher self-efficacy in computer programming tasks than students who do not receive problem-solving and critical thinking instruction.

Sub-Hypothesis One: Students who receive instruction in non-mathematical problem-solving and critical thinking skills will exhibit significantly higher self-efficacy related to their ability to work independently and continue despite difficulty than students who do not receive problem-solving and critical thinking instruction.

Sub-Hypothesis Two: Students who receive instruction in non-mathematical problem-solving and critical thinking skills will exhibit significantly higher self-efficacy related to their ability to perform complex programming tasks than students who do not receive problem-solving and critical thinking instruction.

Sub-Hypothesis Three: Students who receive instruction in non-mathematical problem-solving and critical thinking skills will exhibit significantly higher self-efficacy related to their ability to self-regulate than students who do not receive problem-solving and critical thinking instruction.

Sub-Hypothesis Four: Students who receive instruction in non-mathematical problem-solving and critical thinking skills will exhibit significantly higher self-efficacy related to their ability to perform simple programming tasks than students who do not receive problem-solving and critical thinking instruction.

المنارة للاستشارات

www.manaraa.com

Hypothesis Two: Students who receive instruction in non-mathematical problem-solving and critical thinking skills will exhibit significantly higher achievement than students who do not receive problem-solving and critical thinking instruction.

Research Questions

- What are the primary causes of the declining enrollment of students in computer science?
- What techniques have been implemented to increase student enrollment? What has been the success of these techniques?
- How does specific instruction in non-mathematical problem-solving techniques and critical thinking skills impact the students' computer programming self-efficacy and achievement?

Limitations

1. According to the faculty participating in the study, the attrition rate of the course selected for the study tends to approach 50%. Not all students who began the study remained in the course through its completion. This, in turn, created a small sample size in both the experimental and control group.
2. Programming is an entirely elective course. Students in the selected programming courses self-selected to some degree.
3. Students taking the selected programming classes likely did so because of an interest in the subject matter or external pressure from parents or peers.

www.manaraa.com

4. Random assignment of students to classes was not possible. The study utilized classes that had been formulated by the school and voluntary student enrollment based on schedule preferences.
5. Participation in the study was voluntary. Not every student in the participating classes chose to take part in the study. This contributed to the small sample sizes.
6. Students in the experimental classes did not submit the problem-solving worksheets for each course assignment. These worksheets were intended to show application of the tutorial skills. Since none were submitted, the researcher is unable to determine if they were completed.
7. It was not possible to use a multivariate analysis of the covariance due to small response size. Instead, multiple analyses of covariance were used. This inflates the Type I error rate.
8. There was a large disparity in achievement at the outset of the study between the control and experimental groups. A large gain in achievement through the term by the experimental group, combined with a negligible change in control group scores, led to differences in achievement which were statistically but likely not practically significant.

Definitions of Terms

Computing is an encompassing term used to describe the five distinct computer-based disciplines: Computer Science, Information Technology, Information Systems, Computer Engineering, and Software Engineering (Courte & Bishop-Clark, 2009).

www.manaraa.com

Non-mathematical problem-solving is defined as a generic problem-solving process not related specifically to the process of solving or understanding math word problems. Non-mathematical problem-solving involves techniques that can be applied to a broad spectrum of problems including such things as logic puzzles, logistics, and scheduling (Whimbley & Lochhead, 1999).

Problem-solving is defined as the high order cognitive processes exercised to obtain a solution to a given situation.

Critical thinking is synonymous with problem-solving.

The *think-aloud method of problem-solving* is defined as a method wherein those attempting to solve problems verbalize each step of their mental process either to themselves or to a listening partner. With practice, the think-aloud method can be applied without audible vocalization as the problem solver learns to internalize the thought process (Whimbley & Lochhead, 1999).

The *listening partner* is a passive participant in the think-aloud problem-solving process (Whimbley & Lochhead, 1999).

Bandura (1994) defines *self-efficacy* as an individual's belief in their ability to perform a task.

Computer programming self-efficacy is defined as an individual's belief in their ability to perform computer programming tasks.

Summary

Females continue to be under-represented in computing, with enrollment in the field declining across both genders. Low self-efficacy and deficient problem-solving

المنارة للاستشارات

www.manaraa.com

skills are two factors that have been identified as potential causes for the low enrollment of both genders (Pollock et al., 2004; Olsen, 2005; Paxton & Mumey, 2001; Beaubouef et al., 2001). Males are choosing other careers as a result of a lack of self-confidence, misconceptions about the field, and negative cultural stereotypes as well (Beaubouef & McDowell, 2008; Biggers et al., 2008; Joseph, 2008; Lewis et al., 2008; McInerney, DiDonato, Giagnacova, & O'Donnell, 2006). This study investigated the effect of specific, non-mathematical problem-solving instruction on the computer programming self-efficacy and achievement of introductory computer science students to determine if providing this instruction is one way to begin to reverse the decline in enrollment in computer disciplines.

www.manaraa.com

Chapter 2

Literature Review

Introduction

Academics and professionals have been studying the gender gap in computing for over a decade. Recently, research has expanded to include a study of the decline in enrollment in computing majors for both genders. The majority of the findings indicate that the problem begins with recruiting students into the field (Hart et al., 2008; Hu, 2008; Owens & Matthews, 2008; Rafieymehr, 2008; Rieksts & Blank, 2008; Sands et al., 2008; Van Sickle, 2008) and continues with a need to retain them once they begin a course of study (Barker, McDowell, & Kalahar, 2009; Biggers et al., 2008; Cohoon et al., 2008; Edmondson, 2008; Klawe & Leveson, 1995; Lewis et al., 2008; Lopez et al., 2005; Papastergiou, 2008; Scragg & Smith, 1998; Powell, 2008; Sloan & Troy, 2008; Wilson, 2008).

The primary reasons for the decline in enrollment presented in the research include misconceptions about the field (Beaubouef & McDowell, 2008; Owens & Matthews, 2008; Rafieymehr, 2008, Sands et al., 2008; Sloan & Troy, 2008; Van Sickle, 2008), negative cultural stereotypes (Adya, 2008; Barker et al., 2009; Edmondson, 2008; Sands et al. 2008; Van Sickle, 2008), and lack of self-efficacy (Cohoon et al., 2008; Kumar, 2008; Norris et al., 2008; Pollock et al., 2004; Powell, 2008; Sloan & Troy, 2008). Research indicates that female students encounter additional hurdles that include a lack of computer experience prior to the first year in college (Barker et al., 2009; Hu, 2008; Powell, 2008; Van Sickle, 2008), a paucity of female mentors and role models

www.manaraa.com

(Edmondson, 2008; Powell, 2008), and subtle discriminations in the classroom (Cohoon, Wu, & Chao, 2009; Edmondson, 2008; Sands et al., 2008). Analysis of first year attrition revealed that one of the major challenges faced by students of both genders is a lack of problem-solving skills (Beaubouef et al., 2001; Jin, 2008; Kumar, 2008; Norris et al., 2008; Olsen, 2005; Paxton & Mumey, 2001; Pulimood & Wolz, 2008; Ragonis & Hazzan, 2008). Each of the factors mentioned is represented in both recruitment and retention research.

Misconceptions about the Field

Math Aptitude

One of the common misconceptions about the field of computing is that a high degree of math aptitude is necessary for success (Colley et al., 1996). However, research indicates that the enrollment problem is not due to a lack of aptitude in math or science. Klawe and Leveson (1995) found that female computing students tended to achieve better grades in math and science than their male counterparts. Sackrowitz and Parelius (1996) noted that despite comparable math Scholastic Aptitude Test (SAT) scores, female freshmen in introductory computer courses had significantly lower achievement than males. More recently, Beckwith et al. (2006), Wilson (2006) and Madigan et al. (2007) noted that there was no significant difference in math ability or basic computer skills such as Internet and productivity software use between the genders.

Colley, et al. (1996) found that the perception of math as a necessary skill for computing and the corresponding math anxiety did decrease women’s interest in computing. Scragg and Smith (1998) also researched math anxiety as a possible cause for

المنارة للاستشارات

www.manaraa.com

lack of female retention but they found that, while both men and women recognized the relevance of math for a computing career, both genders expressed a comfort level with math that was not significantly different. Similarly, Nauta and Epperson (2003) found that math and science ability did not have a significant impact on women's computing career choice, instead stating that more significance lay with other variables such as personal interest and perceived job outlook. Wilson (2006) also found no significant difference in math ability between males and females, though a math background was found to have a positive influence on computer science success. Further, Wilson found that, given the choice between a game programming assignment and a math based programming assignment, female students showed a significant preference for the math based assignments. These studies would seem to diminish the importance of math anxiety on interest in computing.

Integrating Personal Interest

Another common misconception of computing is that it is unrelated to interesting problems in other domains. Current educational practices tend to support this misconception by focusing solely on programming languages and software development in introductory classes (Hart et al., 2008; Hazzan, Gal-Ezer, & Blum, 2008). Allan and Kolesar (1997) note that students in introductory computer science courses indicate a singular focus on successful completion of the current assignment with no thought toward the larger picture of how skills learned will apply in various domains or even to other problems within computer science. A variety of research, discussed next, calls for curriculum changes to combat this aspect of the discipline's image.

المنارة للاستشارات

www.manaraa.com

Tang, Pan, and Newmeyer (2008) note that high school girls show considerably more interest and self-efficacy in careers and subjects involving working with and helping people. Rao (2006) noted an increase in performance among female computing students when assigned tasks in a domain of interest. Wilson (2006) also noted that female students were more likely to be interested in computer use when the problems addressed helped serve society as opposed to computer use simply for the sake of using a computer or to discover how it functions.

Klawe and Shneiderman (2005) discuss the importance of a shift in the overall curriculum of computer science to address the use of computers to solve societal problems. Baker, Krause, Yasar, Roberts, and Robinson-Kurpius (2007) state that a perceived lack of societal relevance keeps many students from entering science and engineering majors. Rao (2006) emphasized a need for computer science educators to shift techniques to focus on application in areas of interest rather than theory. Van Sickle (2008) recommends shifting the curriculum to focus more on specializations and certifications needed in industry rather than the broader theory represented by most undergraduate programs. Joseph (2008) recommends a greater incorporation of internships and cooperative education into the curriculum as these have been shown to positively influence the choice of a computing career in both genders. Klappholz (2009) recommends incorporating real-world software engineering projects into computer science classes. These projects would be maintained and modified by future students or students in other courses and the deliverables would be in use by departments at the school, thus providing experience with a real client. Current standards of education

المنارة للاستشارات
www.manaraa.com

involve creation of operating systems and analysis of algorithms in a purely computer science oriented context, devoid of real world application. In addition to helping computer science remain relevant, a shift toward teaching computer science as a mechanism for real world problem-solving may help draw and maintain interest in the subject.

Lau, Ngai, Chan and Cheung (2009) developed a summer camp for students of both genders designed to show the integration of computers with fashion. They used wearable computing and e-textile projects to allow middle school students to discover unusual applications of computer science degrees. Similarly, Owens and Matthews (2008) created a civics curriculum for high school to help demonstrate the importance of computing to the social and political systems in the United States. Cady and Terrell (2007) integrated computer activities into elementary school science activities. All three studies were geared toward encouraging interest in computers and technology by focusing on application to real world problems. The ideas demonstrated by Cady and Terrell are echoed in the Computer Science Teacher's Association's (CSTA) K-12 computer curriculum described by De Clue (2008), which recommends the integration of computers into all subject areas at the elementary level rather than studying computers as a subject of their own.

Negative Cultural Stereotypes

A Masculine Field

Pollock, et al. (2004) report misconceptions, primarily generated through negative cultural stereotypes, held by potential female computer science students include a

المنارة للاستشارات
www.manaraa.com

perception of the field as a masculine field. Takruri-Rizk, Jensen, and Booth (2008) indicate that technology classrooms and exercises tend to center on masculine experiences and learning preferences, lending credence to this misconception. Brown, Garavalia, Fritts, and Olson (2006) investigated the sex role orientations of both male and female students who were majoring in computer science and found no propensity for the female students in computer science to have a more dominant male orientation than average females in this age group. Additionally, sex role orientation was not found to have an influence on achievement in computer science. Despite these findings, the perception remains, with Madigan et al. (2007) reporting that elementary school aged children consider females interested in computer science to be *tomboys* who are less interested in traditionally feminine activities. Lopez, Zhang, and Lopez (2008) suggest that males and females who are more androgynous than masculine or feminine are more likely to be involved in a computing career. They suggest that an emphasis on the androgynous nature of the field may encourage more women to consider it.

Tang et al. (2008) note that women generally avoid careers perceived as male-dominated due to lower self-efficacy, though gifted female high school students did not demonstrate a decline in self-efficacy for male-dominated subjects. Papastergiou (2008) indicates that the perception of computer science as masculine may be changing and reports that the secondary school students in Greece studied did not show a significant belief in the masculinity of the field.

المنارة للاستشارات

www.manaraa.com

Nerds and Geeks

Myers and Beise (2001) previously noted the perception of IT careers as the domain of *nerds* and *geeks*, a perception influenced by the portrayal of computer workers in the popular media. This misconception was also noted by Pollock et al. (2004) who stated that females pinpointed the perception of IT workers as introverted individuals who work on their own with no outside interaction as one deterrent to pursuit of an IT career. Joshi and Schmidt (2006) found that college students listed *nerdy*, *intelligent*, and *lacking in social skills* as three of the primary traits they associated with computer professionals. Even after presentations about the typical work environment for several computer related careers, with special emphasis on the social nature of the work, these characteristics remained prevalent in student descriptions. Edmonson (2008) found that high school seniors of both genders tended to categorize computer science departments as being composed almost entirely of nerdy boys who played video games. This was stated as a primary reason why a computer science major was not something they were considering. Beaubouef and McDowell (2008) also note the common perception of computer geeks in dark rooms alone with their computers, despite the fact that very few computer jobs are solitary in nature. They stress the importance of good written and verbal communication for computer scientists and suggest that educators go out of their way to focus on these facts in an effort to break the stereotype and make the discipline more appealing to students of both genders.

المنارة للاستشارات

www.manaraa.com

Computer Self-Efficacy

Bandura (1994) defines self-efficacy as the belief a person has in their ability to successfully accomplish a particular action. Further, Bandura states that self-efficacy is influenced by a combination of four factors: previous personal experience, observation of others' experiences, effective external motivation, and an individual's emotional response to the task. Self-efficacy begins to develop as newborns start exploring their world and continues through the various phases of growth and development that follow. As children grow, their parents, peers, and schools become increasing influences on their developing self-efficacy. Pintrich and Schunk (1996) state that research demonstrates a decrease in self-efficacy overall as students near middle school. They indicate that research has not yet precisely pinpointed an average age for the decline, noting a demonstrated range that begins as early as the fifth grade and extends to the ninth grade. Liu, Hsieh, Cho, and Schallert (2006) indicate that science and technology self-efficacy begins to decline at age 11. Tang et al. (2008) suggest that by the age of fourteen, children have firmly established preconceptions about careers and self-efficacy. Nauta and Epperson (2003) found that self-efficacy could still be positively influenced in high school and college, particularly if interest and achievement supported persistence in the field. Mayall (2008) found that college students' self-efficacy could be influenced in a positive way and in so doing attitudes toward ability and computers as a career were also positively impacted.

Beckwith et al. (2006) note that first experiences in computing serve to form the foundation for computing self-efficacy and that early failures, or perceived failures, generally set the tone for poor self-efficacy in the future despite future successes.

www.manaraa.com

McInerney et al. (2006) echo this statement, citing research which demonstrates that women, in particular, are more likely to choose a computing career based on positive early experiences, parental occupation, and high school programming courses. High school teachers and counselors were specifically noted as particularly influential in an individual's choice of career. Tang et al. (2008) indicate that female students' self-efficacy is more strongly influenced by a learning experience than their male counterparts. They also note that females tend to choose careers based on an expected outcome more than interest, while males are influenced primarily by interest.

Professional organizations for computer science practitioners such as the Association for Computing Machinery (ACM), the Institute of Electrical and Electronics Engineers (IEEE), and the Computer Science Teachers Association (CSTA) recommend development and implementation of computer science curricula that begin in Kindergarten and continue through high school. Typically these curricula begin with exposure to computers integrated into traditional subjects in the K-8 grade ranges in order to develop literacy and provide positive computer interactions. In grades 9 – 12, the recommendations for courses change to those designed to encourage the development of programming skills and understanding of more formal computer science theory (DeClue, 2008). Cady and Terrell (2007) found that integration of technology into science courses helped bolster the self-efficacy of female students with respect to technology more than providing similar technology exercises in a context unrelated to other subject matter. This technique mirrors the recommendations made by DeClue for computer instruction at the elementary level. Both DeClue and Cady and Terrell suggest that these early positive

المنارة للاستشارات
www.manaraa.com

experiences may encourage female students to pursue further courses in computer science. Rafieymehr (2008) agrees that early exposure to computers is a key element to building self-efficacy and interest in computing. This belief drove the creation of the Generation Link project, which was designed to offer middle school students a chance to work with college students and faculty on hands on computer projects that ranged from hardware labs to programming with Alice. These short, positive exposures show promise for building interest and self-efficacy in middle school girls.

Since self-efficacy is strongly dependent on a particular action, Cassidy and Eachus (2002) emphasize that self-efficacy levels of an individual will vary between domains, thus a person might be highly confident in his or her mathematical ability and still demonstrate low computer self-efficacy. Downey (2006) and Liu et al. (2006) extend this thought, suggesting that even within a specific domain, such as computers, self-efficacy will vary by task because of the same influences. In this situation, an individual may show high self-efficacy in general computer use but still suffer from low self-efficacy when it comes to the solution of programming problems on the computer.

Wilson (2006) confirmed that even in situations where no achievement or skill difference was found between male and female students, female students consistently had lower confidence in their ability with computers. Further, female computer science majors were found to have less confidence in their ability than male non-majors. Cohoon, et al. (2008) and Ballou and Huguenard (2008) both found that women tied perceived performance to self-efficacy. If their grades were not what they expected, women's self-efficacy tended to dip, causing them to consider leaving the program. Cassidy and Eachus

المنارة للاستشارات
www.manaraa.com

(2002) indicated a relationship between the perceived complexity of a task and a gender-based difference in self-efficacy. Females showed a steadily diminishing self-efficacy as computer-based tasks increased in complexity. The self-efficacy of males in the same situation either remained steady or increased slightly as tasks became more complex. Murphy and Thomas (2008) noted this difference as well, stating that female students were more likely to suggest that ability with computers was innate and either possessed or not attainable. Madigan et al. (2007) found that students, regardless of gender, tended to perceive their computer literacy skills (e.g. use of the computer for research and completion of projects) as more developed than they actually were. However, they also found that female students continue to have considerably lower self-efficacy when it comes to completing tasks on a computer, causing female students to hesitate to enroll in computer science classes or take on a computer science major. Those females who do initially enroll are prone to dropping out due to a lack of belief in their ability to complete the degree successfully. According to Goold and Rimmer (2000), females who complete their first course successfully and persist in the program go on to show no difference in achievement from their male counterparts.

Madigan et al. (2007) recommend that instructors in the computer science field find ways to build females' confidence incrementally through the course of their computer science studies but offer no suggestions as to what methods may be beneficial. Quade (2003) suggested that expressing high efficacy expectations motivated students of both genders to persevere in the face of declining self-efficacy. Quade also noted that problem-solving success was a critical component of computer self-efficacy.

المنارة للاستشارات
www.manaraa.com

Lack of Prior Experience

When initially enrolling in a computer course, female students often report less prior experience with computers than their male peers. Moorman and Johnson (2003) cite the propensity for parents to give male children computers as gifts or allow them greater access to the family computer due to their technical nature and a perception that computers are more applicable to males. In many cases, female children are asked to give priority to their brothers when it comes to available computer time. Kiesler, Sproull, and Eccles (2002) note that this trend is also found in school computer labs where male students tend to dominate available computers, leaving little access for females unless strict access control is applied by the teachers.

A typical first introduction to computers at home is through games, most of which target a primarily male audience. The games which specifically target female computer users tend to be less graphically advanced than those targeting males. Even educational software titles were found to be more likely to contain content and story lines that are more traditionally associated with male users, such as shooting and space adventures (Moorman & Johnson, 2003; Madigan et al., 2007). Madigan et al. (2007) found that females tended to use the Internet less frequently than males and that the Internet use that does occur tends to be for the purpose of communication and not entertainment, whereas males spend time doing both. They also noted that males sought time for computer use while females tended to use the computer only when they had a specific purpose in mind before hand.

www.manaraa.com

Ballou and Huguenard (2008) found that students judge their own potential for success in a computer course based on their perception of prior experience, thus it is important that strategies be developed to compensate for lack of experience. One strategy includes instruction in programming concepts through a story-telling visual environment, which reduces the reliance on prior experience for introductory computer education (Moskal et al., 2004). Sands et al. (2008) designed introductory computer classes that teach software such as Adobe Flash. Flash can be used purely as a software application for the creation of multimedia presentations; however it also has a reasonably robust programming aspect for more advanced users. The intent of the course is to generate interest among students first with the application and then draw them into the more complex programming tasks once the usefulness has been experienced.

Mullins, Whitfield and Conlon (2008) introduced Alice, a 3D object-based, drag-and-drop language, in the first of a three course programming series. There were challenges with adapting the first course to use Alice since typical first-year programming exercises do not translate well to the environment. Despite these challenges, students learned fundamental programming principles through this visual experience and had more success with their initial course. This, in turn, significantly reduced the number of students withdrawing from the course sequence. Alice is also captivating for non-majors and the course is functioning as a valuable recruitment tool for the major. Sivilotti and Laugel (2008) employed the Scratch programming language to provide middle school students with an easy to learn, multimedia-based first exposure to programming concepts. Al-Bow et al. (2008) used the Greenfoot IDE, a drag-and-drop

المنارة للاستشارات

www.manaraa.com

environment similar to Alice, to teach introductory Java programming concepts in a summer camp for high school students with no prior programming experience. Students indicated that they gained valuable understanding of these concepts through the visual environment. However, these students were unable to demonstrate knowledge transfer from the visual environment to a paper-based test.

Storytelling can be incorporated into programming courses even without use of a visual programming language. Rao (2006) explored the use of storytelling in computer education and reported increased motivation and achievement in students of both genders. Duvall (2008) developed extensive metaphors that took on the life of fairy tales to explain various programming concepts. The majority of students reported that these stories helped them understand the concept. However, Duvall cautions that not all students will see the meaning behind the story and that the teacher should be aware that all metaphors break down eventually, so it is important to remind students not to focus solely on the stories.

Balch et al. (2008) incorporated robots and the relatively simple scripting language Python as well as multimedia applications into introductory college level classes so that students have almost immediate, tangible success from their programming efforts. Rieksts and Blank (2008) have made similar efforts with the use of robots at the secondary level. Ericson, Guzdial, and Biggers (2007) discuss summer camps and school year workshops conducted for teachers, middle and high school students, and Girl Scout troops. These workshops focus on use of Alice and LEGO Mindstorm robots to teach introductory programming concepts, helping to broaden the experience of potential

المنارة للاستشارات

www.manaraa.com

students in the advanced placement (AP) computer science course and exam. The robots used in the workshops are also made available to qualified teachers for use in their classes, removing a cost obstacle for robot integration into the classroom.

Sloan and Troy (2008) recommend retooling the typical college computer science sequence to create a CS 0.5 course designed to work with both male and female students who have no prior experience. They recommend that the new course be considered mandatory for all students so that there is no stigma associated with taking it. Students with considerable prior experience can choose to take a placement exam for the CS 1 course should they desire to do so. The CS 0.5 course uses multimedia to introduce students to computer science and help develop what Sloan and Troy refer to as *programming maturity*. Initial results have shown that participation in the new course has increased retention and achievement for future computer science courses.

Hardy (2008) recommends shifting the focus of high school computer science courses from traditional programming instruction to the use of Web 2.0 technologies such as blogs and wikis. These technologies are considered more interesting to high school students and are still able to introduce elementary programming concepts. In addition, it is reasonably simple to incorporate a student's individual area of interest into a generic assignment that uses Web 2.0 technology. So students are not all creating the same program, but rather they are programming projects that tie into something in which they are personally interested.

www.manaraa.com

Lack of Female Mentors and Role Models

A variety of studies have evaluated the benefit of same-sex education, either during primary technology education or in supplemental summer camp situations for pre-college girls (e.g. Adams, 2007; Doerschuk, Liu, & Mann, 2007; Hu, 2008; Maloney, Peppler, Kafai, Resnick, & Rusk, 2008; Olivieri, 2005; Pollock et al., 2004). These programs are designed to take place prior to the girls arriving at college as studies have shown that a large number of girls have decided against computing long before they graduate from high school. Same-sex education programs seem to generate increased interest in computing among the girls participating. Additionally, they provide an opportunity for the girls to receive mentoring from women in the field, network with other girls their age who have an interest in computing, and have increased access to computer resources since they do not need to compete with male students for computer time. Not only do these programs address the specific lack of mentorship identified as an influence in the gender gap and provide an opportunity for girls to build experience on computers, but they help combat the feelings of isolation and frustration frequently experienced by women in computing (Olivieri, 2005; Sackrowitz & Parelius, 1996).

Adams (2007) discusses a computer camp for middle school girls. This age group was chosen due to findings that indicated that by high school, girls have already associated computer science with various common stereotypes and removed it from their mental list of acceptable interests. Middle school aged girls were found to still be open to the discipline, despite the fact that Lee (2008) cites multiple research findings indicating that children have begun to form beliefs about careers as early as age five. Adams found

www.manaraa.com

that a two-week, single-gender summer camp that introduced programming concepts generated considerable interest in computing among the participants. The 3D virtual world, Alice, that was used to introduce the programming concepts was deemed motivational and also encouraged socialization and peer mentoring.

The summer camp described by Hu (2008) also uses Alice to introduce programming in a 3D, drag-and-drop format. While the camp offers other technology workshops, the Alice workshop is consistently one of the most popular. The middle school girls attending this camp are also given the opportunity to interact with female computer science faculty and students from the sponsoring college. Hu intends to study the long-term impacts of camp attendance as the first groups of girls move through their high school years.

Doerschuk et al. (2007) explored a one-day camp format for middle school girls that focused on brief introductions to a variety of computer-based concepts. Each activity was designed to demonstrate the usefulness of computers in society and provide a positive and confidence-boosting experience. In addition to these factors, the one-day format was chosen to reduce costs and make the camps accessible to a broader audience. Feedback from the participants indicated that even the small exposures to the various computing technologies possible in a one-day camp were beneficial in revising opinions about computing as a career.

Maloney et al. (2008) describe a community technology center designed to give underprivileged kids a chance to work with computers after school. A multimedia introductory programming language called Scratch was installed on the machines in the

المنارة للاستشارات
www.manaraa.com

community center. Like Alice, Scratch lets users “program” by dragging objects from libraries onto a stage. These objects can then be animated in various ways to create things like movies and games. Though there was no set curriculum for using Scratch in the clubhouse, it rapidly became one of the most popular software programs for both genders. The overwhelming popularity of programming with Scratch in this afterschool program suggests that interest in computers as a whole has also been improved, though no studies have yet been done at the center to confirm this. Sivilotti and Laugel (2008) also used Scratch in a three-hour programming workshop that is part of a week-long computer science camp for middle school girls. Success with this technique was demonstrated in increased enrollment in this workshop after the initial offering and survey results indicating that attitudes had been changed to be more favorable to computer science as a result of their experience.

Other techniques for mentoring have also been the subject of considerable research. Takruri-Rizk et al. (2008) notes that a single teacher or experience is generally not enough to provide the mentoring needed to encourage female students; family members working in technology and encouragement by peers is also critical. Townsend, Barker, Menzel, and Cohoon (2008) organized regional women-only conferences. These conferences were designed to help women meet others in similar fields nearby and allow them to network and share ideas. This, in turn, can help dispel feelings of isolation that many women in computing fields experience. Townsend et al. note that in addition to achieving these goals, participants began several collaborative projects for further networking and research.

المنارة للاستشارات

www.manaraa.com

In a similar vein, college campuses are beginning to recognize the need for female computer science students to find support. Beck (2007) notes that female freshman computer science majors struggle with the male dominated environment in the classroom. This environment includes a propensity for male students to refer to themselves as "geeks" or "hackers", terms with which women do not typically identify. In addition, female students tend to evaluate their performance lower than a male student of the same ability. Beck formed a support group for female computer science students to attempt to combat these problems. Participants in the support group had structured and scheduled time with faculty mentors and other female majors. In addition, the group went on several field trips to see computer science in action in the workplace. Students who participated actively in the support group were significantly more likely to graduate in computer science than those who opted not to participate.

Powell (2008) reports that not all women feel comfortable joining a support group, despite evidence that it is beneficial. Students indicated a perception that participation in a support group meant they were less capable than their male counterparts, particularly since there was not a corresponding male support group. Several students who initially dismissed the idea of the group eventually joined and expressed their appreciation for the mentoring and socializing that the group provided, crediting the group with their decision to continue in the major after their freshman year.

Cohoon et al. (2008) found that women who did not feel comfortable asking questions in class were significantly more likely to leave the major. A single-gender study and support group was found to alleviate this trend to some degree.

المنارة للاستشارات
www.manaraa.com

Classroom Discrimination

Treu and Skinner (2002) indicate that discrimination against female computer science students is characterized by several behaviors: females are called on by name less often, are interrupted more often, and given less time to answer questions than their male peers. Online asynchronous learning environments would seem to provide a straightforward solution to these gender inequities because all students must be addressed by name when interacting in a written medium, interruptions are impossible in an asynchronous environment, and every student has an equal amount of time to respond to questions in an asynchronous environment. However, the differences between the language used by males and females carries over into the online environment and can still lead to gender discrimination that even the use of gender neutral pseudonyms cannot circumvent (Guiller & Durndell, 2007). Carr, Cox, Eden, and Hanslo (2004) found that males tended to dominate online discussions and were prone to ridiculing females who attempted to participate. This online bullying was found to be more pronounced than observed in face-to-face situations.

Edmonson (2008) notes that high school girls veered away from further computer science courses after experiencing negative treatment at the hands of their male peers. Sands et al. (2008) noted similar behaviors but suggests that using Flash to teach programming combats some of the discriminatory behavior between students since generally all students begin with no prior knowledge when this technology is employed.

The problem of discrimination continues into graduate programs and the corporate world. Cohoon et al. (2009) found that graduate programs and careers where

المنارة للاستشارات
www.manaraa.com

men outnumber women, such as computer science programs, are more likely to turn a blind eye to sexist remarks and instead expect women to recognize these remarks as harmless and acceptable. Though this environment can fall short of a legal definition of harassment, it has been shown to increase the number of females who withdraw from the program or change careers.

Problem-solving and Critical Thinking

Problem-solving skills are a critical component of success in computer science. This fact is recognized by practitioners and the majority of students who have made it to the second year of a computer science major, but it is not something addressed directly in typical computer science texts and classrooms (Beaubouef & McDowell, 2008; Biggers et al., 2008; DeClue, 2008; De Palma, 2001; Eastman, 2003; McInerney et al., 2006; Sullivan & Lin, 2006; Teague, 2002). Goold and Rimmer (2000) found that problem-solving ability is one of the most critical indicators of first year college computer majors' success. Several researchers have identified problem-solving as one of the primary activities in computing disciplines enjoyed by women (Colley et al., 1996; De Palma, 2001; Joiner et al., 1996; Klawe & Schneiderman, 2005; Rao, 2006; Teague, 2002).

Cho (1995) studied the use of LogoWriter as a method for teaching critical thinking in the context of an introductory computer science course. The study concluded that focusing on the critical thinking aspect of programming helped reduce computer anxiety and increase the self-confidence of students of both genders in terms of their technological abilities. Colley et al. (1996) suggested that redefining computing as a problem-solving oriented discipline, rather than a mathematically based one, may help

المنارة للاستشارات

www.manaraa.com

combat women's misperceptions of computing. The suggestions of Klawe and Schneiderman (2005) and Rao (2006) are extensions of these early recommendations for emphasizing the problem-solving nature of computer science in order to facilitate student achievement and interest.

Gibson and O'Kelly (2005), Olsen (2005), Eastman (2003), Joyce (1998), and Tu and Johnson (1990) have investigated ways to use the computer as a problem-solving tool for non-majors and to teach computer science majors a four-step problem-solving method based on the 1948 work of Polya. This method is essentially the same for each researcher and can be summarized as understanding the problem, creating a plan for solving the problem, implementing that plan, and verifying the solution (Tu & Johnson, 1990). This basic sequence is, for computer science majors, then translated into a simple approach to software engineering with the sequence of steps becoming: determine the input and output of the program; design the program using flowcharts, algorithms or pseudo code; write the program; and debug and test the program (Olson, 2005; Joyce, 1998; Tu & Johnson, 1990).

Eastman (2003) recommends focusing specifically on breaking the problem into discrete facts as the first step of the problem-solving sequence (understand the problem). In his experience, Eastman feels that students learn and follow the sequence without truly understanding the problem or the steps they took to arrive at the solution. A proper understanding of the facts of the problem and a determination of their relevancy to the solution is, he feels, the most critical component. Lai and Wong (2007) recommend the use of various diagramming techniques such as force field analysis and fishbone

المنارة للاستشارات

www.manaraa.com

diagrams to help introductory programming students break down problems. Jin (2008) found that guided analysis of the problem at the start of programming assignments that helped students identify inputs, processes, and outputs with the problem statement assisted students in understanding basic concepts more quickly. Ali (2005) emphasizes the need for computer science instructors to teach students the principles of critical thinking and problem-solving as part of introductory computer classes. In addition, Ali states that students must be explicitly taught how to make connections between concepts. Ragonis and Hazzan (2008) echo the recommendations of Ali, suggesting that teachers use a Socratic method to help students improve their problem-solving strategies. Olsen (2005) recommends using pseudo code to take the focus off actual implementation of the solution in a particular programming language and refocus students on the problem-solving portion of programming before gradually introducing them to syntax and other language-specific aspects of an introductory computer science course. Kumar (2008) suggested that providing additional practice problems beyond the primary programming assignment was necessary. These practice problems were analyzed by an online tool and feedback returned to the student. Students who participated in these supplemental exercises did appear to gain self-confidence, even if achievement was not impacted.

Gibson and O'Kelly (2005) as well as Ali (2005), Reed (2002) and Daigle, Doran and Pardue (1996) recommend using groups in computer science courses during the initial design stages of each programming assignment (the problem-solving stage). Pulimood and Wolz (2008) stress the need for computer science courses to focus on not only individual problem-solving skills but on an ability to solve problems in groups, as

المنارة للاستشارات

www.manaraa.com

this is more representative of the typical work environment encountered after graduation. Falkner and Palmer (2009) build on the recommendations of Eastman (2003) and Ali (2005) and recommend the incorporation of problem-solving lectures, where students observe problem-solving activities, followed by collaborative problem-solving exercises. Falkner and Palmer feel that the collaborative aspect of problem-solving is particularly relevant to real world applications of computing and cannot be omitted. Arshad (2009) also had students observe problem-solving activities, though he encouraged students to ask questions during the process while Falkner and Palmer used more of a lecture format for those observations. Unlike Falkner and Palmer, Arshad did not follow these observations with individual practice in problem-solving, relying instead on the students' initiative to practice the techniques demonstrated. Despite this, Arshad noted considerable increases in achievement with these demonstrations. Hanks and Brandt (2009) used pair programming to study the problem-solving methods used by students programming in Java and found that very few spent time ahead of the initial coding stage determining the nature of the problem or evaluating what steps should go into the solution. They recommend teachers focus on helping students understand the necessity of this initial design and problem-solving process.

Paxton and Mumey (2001), Allan and Kolesar (1997), and Beaubouef et al. (2001) describe the experimental programs tested at their universities to include problem-solving instruction in computer science courses. Paxton and Mumey (2001) integrated problem-solving requirements into their advanced algorithms course. Their focus was on high-level math-oriented problem-solving techniques and specific programming

المنارة للاستشارات
www.manaraa.com

assignments that highlighted these techniques, with a goal toward improved performance in programming contests. While the authors labeled the experiment a success, they also indicated that success at the programming contest was dependent more on teamwork skills and speed than on any of the specific problem-solving techniques learned in the course. Hart et al. (2008) also recommended a return to math-oriented problem-solving techniques in introductory computer courses. Their focus on concepts from discrete math such as Boolean AND, OR, and NOT showed some success in helping students better understand logic needed in programming assignments.

Allan and Kolesar (1997) used typical logic puzzles and games to foster the set of nine problem-solving skills they identified prior to the course, with a focus on self-talk during problem-solving activities. They found that students who took the problem-solving course prior to the initial programming course achieved a minimum of one letter grade higher in the programming course than students who did not take the problem-solving course. Beaubouef et al. (2001) followed a strategy similar to Allan and Kolesar (1997), providing logic puzzles and games in a computer-based format that provided hints or the solution as needed as well as an explanation of how the student should have arrived at the solution.

Beckwith et al. (2006) investigated the correlation of achievement and self-efficacy to the typical reticence of female students to tinker. Tinkering behaviors were classified as problem-solving skills, thus the resistance to tinkering displayed by female students indicates a lack of one subset of problem-solving skills necessary for success in programming. Baker et al. (2007) noted a similar lack of tinkering in female students and

المنارة للاستشارات
www.manaraa.com

suggested that bugs need to be presented as opportunities for problem-solving rather than programming defects. This change in terminology and presentation encourages females to tinker and attempt to fix the bugs rather than embrace feelings of defeat.

Lemire (2002) highlights the lack of research supporting the widely accepted idea that problem-solving skills learned in one discipline, such as math, can be easily transferred to other disciplines without specific instruction. Lemire provides examples demonstrating the lack of transferability of these skills and suggests that while some students may be able to make the transfer on their own, for skills to universally be applicable in a particular domain they will need to be taught in that domain. These findings echo the previous findings of Palumbo (1990) who provided a comprehensive review of the literature available at the time and found little support for the efficacy of using a computer to teach problem-solving capabilities. This review also found little support for the transferability of problem-solving skills from other domains, such as math, to success in solving computer programming based problems. Palumbo further stated that the high school population is one noted for a lack of problem-solving skills in general and that this lack would have a far reaching impact on their abilities and self-efficacy in many different domains.

Summary

This literature review focuses on the primary factors identified in current research for the decline in enrollment in computer science, beginning at the secondary level and extending into the professional marketplace. Each of these factors has been the subject of

www.manaraa.com

considerable research. No clear solutions have yet been identified, though several suggestions have met with small successes.

Female students often demonstrate a lack of prior computer experience when entering their first computer programming course. This is in contrast to their male peers who have generally been tinkering with computers since they were first allowed to do so. To overcome this lack of experience, researchers have explored the creation of female-oriented computer games, using storytelling in the classroom rather than traditional lectures, transitioning to multimedia or 3D drag-and-drop programming environments for first courses rather than the more traditional programming languages, and using robots to provide a physical manifestation of the programs created rather than relying on screen-based output (Al-bow et al., 2008; Balch et al., 2008; Ballou & Huguenard, 2008; Duvall, 2008; Ericson et al., 2007; Madigan et al., 2007; Moorman & Johnson, 2003; Moskal et al., 2004; Mullins et al., 2008; Rao, 2006; Rieksts & Blanks, 2008; Sands et al., 2008; Sivilotti & Laugel, 2008).

Other researchers suggest that the lack of prior experience is best overcome by making modifications in the classroom that range from conscious enforcement of equal time for students of both genders to changing the first course of college majors to one that assumes no prior experience (Kiesler et al., 2002; Sloan & Troy, 2008). Hardy (2008) recommends shifting high school computer science courses away from programming entirely and focusing on technologies of interest such as blogs and wikis instead. While researchers agree that this might be effective short-term, many reiterate the fact that programming is an essential part of computer science, whether or not a career of

المنارة للاستشارات

www.manaraa.com

programming is the end goal (Beaubouef & McDowell, 2008; DeClue, 2008; Eastman, 2003; Gibson & O'Kelly, 2005). Ballou and Huguenard (2008) note that students who have previously learned a programming language perform better than students with no programming experience even in courses that require no programming.

Misconceptions about the field of computer science are common amongst both genders, despite the prevalence of computers in modern life. These misconceptions range from the belief that one must have great ability in mathematics in order to succeed with computers, to the idea that computer science is favored only by socially inept males. Researchers have found that math aptitude is not an indicator of success in computer science (Beckwith et al., 2006; Madigan et al., 2007; Wilson, 2006). Recent research suggests that math anxiety is becoming less of a factor in the decision to pursue a career in computers (Nauta & Epperson, 2003; Wilson, 2006). Brown et al. (2006) found no dominance in gender orientation for computer science students and Lopez et al. (2008) found an indication that a more androgynous oriented person, be they male or female, was more likely to choose computing. Tang et al. (2008) and Takruri-Rizk et al. (2008) suggest that the perception of male dominance comes from the typical classroom exercises, examples, and teaching methods used in a computer classroom. Edmonson (2008) and Beaubouef and McDowell (2008) note the perception of computer scientists as socially inept loners and recommend that the social nature and importance of communication skills become a more prominent part of computer education.

Many other researchers recommend the integration of computer courses into other problem domains that are more traditionally of interest to a broader range of students or

المنارة للاستشارات
www.manaraa.com

creating assignments that represent these interests (Allan & Kolesar, 1997; Joiner et al., 1996; Klappholz, 2009; Klawe & Schneiderman, 2005; Rao, 2006; Tan et al., 2008; VanSickle, 2008; Wilson, 2006). Single gender education is recommended by several researchers as a mechanism for providing an atmosphere where females are not embarrassed to succeed with computers. This suggestion is typically accompanied by recommendations for implementing same gender mentoring programs (Adams, 2007; Beck, 2007; Doerschuk et al., 2007; Hu, 2008; Maloney et al., 2008; Olivieri, 2005; Pollock et al., 2004; Powell, 2008). Edmonson (2008), Cohoon et al. (2008), Sands et al. (2008) and Treu and Skinner (2002) researched ways to remove discrimination from classroom environments to encourage participation of females.

Studies of computer self-efficacy have found that females consistently report lower confidence with computing tasks than their male peers despite not having any significant difference in actual ability. These same studies have found that male students tend to have a higher sense of their ability than is actually borne out in practice. Researchers disagree about the age at which the decline in self-efficacy begins, though the general range appears to be somewhere between the ages of 9 and 14 (Liu et al., 2006; Tang et al., 2008; Pintrich & Schunk, 1996). This lower self-efficacy has been tied to a propensity to withdraw from computer majors as the material becomes more complex and that complexity lowers female self-efficacy further. This phenomenon has been seen to some degree with male students as well. Other researchers suggest that self-efficacy can be positively influenced beyond that age range, regardless of previous experiences (Mayall, 2008; Nauta & Epperson, 2003). Researchers agree that ways must

المنارة للاستشارات

www.manaraa.com

be found to increase female self-efficacy if there is going to be a reversal in the computing gender gap. Primary suggestions for increasing self-efficacy center around creating positive early experiences with computers (Beckwith et al., 2006; DeClue, 2008; Cady & Terrell, 2007; Cassidy & Eachus, 2002; Goold & Rimmer, 2000; Madigan et al., 2007; Quade, 2003; Wilson, 2006).

Problem-solving has been identified by a number of researchers as a critical component of success in the computing discipline despite not being a topic addressed directly in the majority of computer science text books and classrooms (Beaubouef & McDowell, 2008; Biggers et al., 2008; DeClue, 2008; De Palma, 2001; Eastman, 2003; McInerney et al., 2006; Sullivan & Lin, 2006; Teague, 2002). Typically, computer science programs assume students have learned to problem solve in prior math instruction. Further assumptions are made which suggest this prior knowledge will transfer to computer programming problem-solving. Research by Lemire (2002) and Palumbo (1990) suggest that this transfer does not routinely occur.

Some previous studies incorporating problem-solving instruction into programming courses at the college level have shown positive results in terms of improving achievement. Researchers recommend various strategies for specifically teaching students programming oriented problem-solving techniques (Ali, 2005; Allan & Kolesar, 1997; Beaubouef et al., 2001; Eastman, 2003; Hart et al., 2008; Jin, 2008; Lai & Wong, 2007; Paxton & Mumey, 2001; Ragonis & Hazzan, 2008). Each of these studies showed an improvement in achievement by students mastering the techniques; however none of the studies specifically studied the impact on self-efficacy. Kumar (2008) found

المنارة للاستشارات
www.manaraa.com

that students who did additional exercises gained self-confidence through the experience, but they did not receive specific instruction in problem-solving in addition to the supplemental problem sets.

Problem-solving has been identified as an aspect of computing that is particularly interesting to women (Colley et al., 1996; De Palma, 2001; Joiner et al., 1996; Klawe & Schneiderman, 2005; Rao, 2006; Teague, 2002). Baker et al. (2007) found that female students' self-efficacy dropped when they achieved a lower grade than they desired. Since problem-solving instruction has been demonstrated at the college level to increase interest and achievement, the question arises what effect this instruction would have on the self-efficacy of female students. Given that positive early mastery experiences are more beneficial to the development of self-efficacy, this study has investigated the impact of specific problem-solving instruction on the self-efficacy of introductory students.

www.manaraa.com

Chapter 3

Methodology

Introduction

This study investigated whether specific instruction in non-mathematical problem-solving techniques and critical thinking skills impacts the computer programming self-efficacy and achievement of introductory computer science students. It was conducted over a standard 16-week college semester beginning January 10th, 2011 and ending April 29th, 2011. Classes chosen to participate in the study were formed prior to the initiation of the study. This chapter will describe the instructional treatment in more detail.

Study Design

This study was conducted using a quasi-experimental pretest-posttest control group design. This design was chosen because it most adequately controls for all sources of internal invalidity. There was minimal risk of pretest-posttest interaction as the study was conducted over the course of approximately four months. The preferred timeframe for the study was the first semester of the school year in order to minimize any potential impacts of prior instruction and to provide as level a baseline of knowledge as possible. However, the spring semester was deemed acceptable as the course used did not require prior experience in programming.

Two coed groups of students were presented with a course in introductory computer science that focused on introductory programming skills using the Visual Basic programming language. Each group took the *Computer Programming Self-Efficacy Scale*

www.manaraa.com

(CPSES) prior to the beginning of instruction. The control group completed the course having been taught the programming unit using traditional pedagogical methods. The experimental group received additional instruction in problem-solving techniques that were geared toward developing the critical thinking skills necessary for solving programming problems via an online tutorial (see Appendix B) in addition to the programming instruction. The *CPSES* was taken again in the middle of the semester and for a third time at the end of the term. The results of these tests were evaluated to determine if there is a significant difference in achievement and self-efficacy among the students. This test data and results are provided in more detail in Chapter 4 and Chapter 5.

Students participating in the study were enrolled in an introductory programming class at a community college in South Florida. The researcher has no affiliation with this college. Five instructors scheduled to teach the selected course were invited to participate in this study. Three instructors agreed. Two instructors, teaching one section each, were assigned to the experimental group. The third instructor, teaching two sections, was assigned to the control group. The control group instructor requested to be the control group due to a desire to assist in the study but a concern over having any additional duties. At the start of the term there were 71 students in the experimental group and 31 in the experimental group.

Students self-select into computer science courses at the school because they are elective in nature. This self-selection should not create sampling bias as there is no population that could be used for this study that would not be composed of self-selecting

المنارة للاستشارات

www.manaraa.com

students. The first opportunity for students to study computer programming in a formal environment is the best available option from which to select the sample population when looking at self-efficacy. An introductory college programming course is often the first exposure to actual programming instruction, though students may have been previously exposed to other aspects of computer use. The literature recommends evaluating self-efficacy as specifically as possible. This recommendation indicates that technology self-efficacy is different from computer literacy self-efficacy which is different from computer programming self-efficacy (Bandura, 1994; Cassidy & Eachus, 2002; Downey, 2006; Liu et al., 2006). It is possible that declining technology self-efficacy, based on prior non-programming experiences with computers, has already convinced some students not to attempt a computer programming course. However, this study was not designed to evaluate a technique for recruitment into the subject area, but rather one geared toward retention. A significant sampling bias was not anticipated since there is no available population where programming students have not self-selected. The results should, therefore, be transferable to other similar student populations.

Several studies have tied problem-solving skills to an improvement in achievement within various levels of programming instruction (Ali, 2005; Allan & Kolesar, 1997; Beaubouef et al., 2001; Eastman, 2003; Hart et al., 2008; Jin, 2008; Lai & Wong, 2007; Paxton & Mumey, 2001; Ragonis & Hazzan, 2008). Others (Baker et al., 2007; Beckwith et al., 2006; Kumar, 2008; Teague, 2002) have indicated a link between self-efficacy and achievement. Because of these links, in addition to administering the CPSES, faculty provided student achievement data at the mid-point and end of the term.

www.manaraa.com

Instrumentation

Prior to the start of the study, students in both the control and experimental groups were invited to take the *CPSES* via an online form (Ramalingam & Wiedenbeck, 1998). This scale was used to measure student computer programming perceived self-efficacy on four factors deemed critical to success in computer programming: independence and persistence, ability to perform complex programming tasks, self-regulation, and ability to perform simple programming tasks. The *CPSES* has been shown by Ramalingam and Wiedenbeck (1998) to have an overall Cronbach's *alpha* reliability of 0.98. The *alpha* reliabilities for the four factors, in the order listed previously, were 0.94, 0.94, 0.86, and 0.93. These values are acceptably high for use in this study. Students were provided a link to an online copy of the *CPSES* and were asked to complete the 7 point Likert scale 31-question survey within the first week of class. The knowledge and skills evaluated by the *CPSES* are congruent with the topics covered in the programming classes during the timeline of the study. The *CPSES* was administered in an identical fashion three times: at the start of the study, at the midpoint, and at the end.

Procedures

Pre-Study

Approvals from the Institutional Review Board (IRB) at Nova Southeastern University and the college where the study took place were obtained prior to the start of the study. The researcher contacted participating teachers prior to the start of the term. As part of this communication, links to materials and consent forms were provided and instructor questions were answered.

المنارة للاستشارات

www.manaraa.com

Week 1 (January 10th – 16th, 2011)

During the first week of the study a brief explanation of the study was given to students by the instructors both verbally and in writing. In order to maintain anonymity of participants, the instructor of each class assigned each student with a study number. This number consisted of the school assigned course registration number followed by a dash and a number from one to the number of students in that section. (For example, 11460-01 would correspond to student one in the 11460 section of the course.) The instructors of the course maintained the only list linking study numbers to students. Any information given to the researcher by student or faculty relied on the use of study numbers only.

Participating students in both the experimental and control groups took the *CPSES* as a pre-test (see Appendix A). Students in the experimental group were subsequently provided access to the online problem-solving tutorial and encouraged to complete the tutorial as soon as possible. After completing the tutorial, students submitted a web-based form with their study ID acknowledging completion of the five tutorial lessons. The tutorial remained available to students for review for the duration of the study, however all participating students completed the tutorial and submitted the form indicating this completion during the first week of the study.

In this tutorial, students were taught how to use a method of problem-solving called *think-aloud*, which helps them learn to work through problems verbally with a listening partner as a step toward being able to complete problem-solving steps without vocalization. The think-aloud method of problem-solving was developed by and shown to be effective by Whimbley and Lochhead (1999). Students were also given instruction

www.manaraa.com

about distinguishing relevant information from irrelevant information in a programming problem statement and categorizing the relevant information as an input, output, or process to be performed by the final program. Students were then shown how to take the classifications they created and use them in the design of a programming solution. These programming domain specific problem-solving steps are similar to those recommended by many researchers (e.g. Eastman, 2003; Jin, 2008; Lai & Wong, 2007) and are a clear translation into the programming domain of the problem-solving method first recommended by Polya in 1948. The experimental classes incorporated an introduction to the think-aloud method that included collaborative exercises. In the tutorial, students were also presented with an introduction to problem breakdown and classification, and encouraged to incorporate these ideas into assigned exercises.

Weeks 2 – 7(January 17th – February 20th, 2011)

Students in both groups continued with programming instruction as per the standard curriculum. Students in the experimental group were asked to complete and submit a worksheet (see Appendix B) that reinforced the concepts from the problem-solving tutorial as part of each programming assignment. Though the worksheets were not optional and the faculty participating in the study agreed to require these worksheets, no worksheets were received for the duration of the study by the researcher. Interactions with the instructors of the experimental group indicated that students were being reminded and encouraged to complete the worksheets as part of the experimental class instruction, but that students were not following through on this aspect of the study.

المنارة للاستشارات

www.manaraa.com

Week 8 (February 21st – 27th, 2011)

Students in both groups continued with programming instruction as per the standard curriculum. Both groups of students also completed the mid-term *CPSES* survey online. Students in the experimental groups were asked to complete and submit a worksheet that reinforced the concepts from the problem-solving tutorial as part of each programming assignment. Though the worksheets were not optional and the faculty participating in the study agreed to require these worksheets, no worksheets were received for the duration of the study by the researcher.

Weeks 9 – 15 (February 28th – April 11th, 2011)

Students in both groups continued with programming instruction as per the standard curriculum. Students in the experimental group were asked to complete and submit a worksheet that reinforced the concepts from the problem-solving tutorial as part of each programming assignment. Though the worksheets were not optional and the faculty participating in the study agreed to require these worksheets, no worksheets were received for the duration of the study by the researcher.

Week 16 (April 11th – 17th, 2011)

Students in both groups continued with programming instruction as per the standard curriculum. Both groups of students also completed the final *CPSES* survey online. Completion of the final survey was the last activity of the study. When final grades had been calculated, the participating instructors provided the researcher with midterm and final grades for each student in their class who completed all three surveys.

المنارة للاستشارات

www.manaraa.com

Resources and Budget

All students had access to a computer with an Internet connection for the completion of the online CPSES. Students in the experimental group had additional access to a computer with an Internet connection for the completion of the problem-solving tutorial in the first week and the submission of the brief worksheet each following week.

Budget requirements for this study were minimal. The tools and materials used were already in place within the participating schools. Some minor costs for web-hosting were incurred by the researcher.

Research Personnel

Research personnel involved in this study included the researcher and her dissertation committee. Instructors of the participating classes also participated by providing students with researcher-supplied information and Internet addresses for the CPSES survey and problem-solving tutorial. Instructors of the experimental classes provided time for survey completion during their course.

Milestones

The following section provides a more concise summary of the study process detailed above. The study took place from January 10th, 2011 to April 17th, 2011. Prior to January 10th, participating instructors were contacted and the procedures for the study fully explained and agreed to. The following shows the time table for the entire study.

www.manaraa.com

Table 1 *Research Study Timeline*

Timeframe	Activities
Week 1 (January 10th – 16th, 2011)	• Instructors explained the study to their students. • Students were given their assigned study ID numbers for use on all study materials. • Students in both groups took *CPSES.* • Students in the experimental group took the online problem-solving tutorial and submitted a verification of completion form.
Weeks 2 – 7 (January 17th – February 20th, 2011)	• Students in both groups participated in class. • Students in the experimental group were asked to submit an additional problem-solving worksheet with each assignment. • The researcher communicated with experimental instructors to ascertain why no problem-solving worksheets were submitted. Participating instructors

المنارة للاستشارات

www.manaraa.com

	were unclear why worksheets were not being completed. Additional announcements and reminders were made to the classes.
Week 8 (February 21st – 27th, 2011)	• Students in both groups participated in class. • Students in both groups retook the *CPSES*. • The researcher and experimental faculty continued to communicate regarding worksheets. Instructors encouraged students to submit their additional problem-solving worksheets.
Weeks 9 – 15 (February 28th – April 10th, 2011)	• Students in both groups participated in class. • The researcher and experimental faculty continued to communicate regarding worksheets. Instructors encouraged students to submit their additional problem-solving worksheets.
Week 16 (April 11th – 17th, 2011)	• Students in both groups participated in class.

المنارة للاستشارات
www.manaraa.com

	• Students in both groups retook the *CPSES*. • Instructors provided the researcher with midterm and final grade information for all students who completed the study.

Summary

This study examined the impact of explicit non-mathematical problem-solving instruction on the computer programming self-efficacy and achievement of college students enrolled in introductory computer science classes. The study took place over the Spring 2011 term. The major milestones for the study were described in this section.

Students in both the experimental and control classes took the *CPSES* at the beginning of the study, in the middle of the study, and at the end of the study. Students in the experimental classes were given access to an online problem-solving tutorial and asked to complete a worksheet with each programming assignment. The self-efficacy test scores were analyzed to see what impact, if any, the problem-solving instruction had on a student's perception of programming self-efficacy and achievement. It was anticipated that the students in the experimental group would have significantly higher self-efficacy and achievement than students in the control group. These results are discussed in detail in Chapter 4.

المنارة للاستشارات

www.manaraa.com

Chapter 4

Results

Introduction

The *CPSES* instrument measures programming self-efficacy by considering four factors: independence and persistence, ability to perform complex programming tasks, self-regulation, and ability to perform simple programming tasks (Ramalingam & Wiedenbeck, 1998). Students were asked to rank 31 questions on a Likert scale of 1 to 7. Ramalingam and Wiedenbeck defined the 7-point Likert scale for the *CPSES* in this manner: a score of one is “not confident at all”, two is “mostly not confident”, three is “slightly confident”, four is “50/50”, five is “fairly confident”, six is “mostly confident”, and seven is “absolutely confident”. These were the definitions used in this study. This chapter analyzes the results of the pre-test, mid-test, and post-test surveys, evaluating each of the four self-efficacy factors individually. Finally, mid-term and final achievement data is analyzed.

Data Analysis

Ideally, a multivariate analysis of the covariance would be used to analyze the pre-test, mid-test, and post-test data (Gay, Mills & Airasian, 2006), but due to the small sample sizes this was not possible. Instead, student responses to the questions relating to each of the factors were analyzed individually (one for each factor) using an analysis of covariance (ANCOVA). An ANCOVA was also used to analyze the mid-point and final achievement data for significance.

www.manaraa.com

Demographics and Prior Experience

The number of responses to each survey declined as the study progressed. The participating faculty indicated that this was expected as they are accustomed to seeing significant attrition from this course. Table 2 shows a breakdown of the number of survey responses received on the pre-test, mid-test, and post-test for each group by gender.

Table 2

Survey Response Demographics

Group	Pre-Test		Mid-Test		Post-Test	
	Male	Female	Male	Female	Male	Female
Experimental	51	11	34	8	26	8
Control	24	7	14	5	16	5

Only the data from students who participated in all three surveys was useful for analysis. The total number of students who took all three surveys is 43. Table 3 shows the breakdown of the responses from students who participated in all three surveys by group and gender. Additionally, Table 3 shows the subset of those final totals who reported prior experience with programming on the pre-test. The experience reported by students in the experimental group varied from previous and extensive experience with Visual Basic (the language used in the study course) to web programming in HTML. The experience indicated by students in the control group primarily indicated web

المنارة للاستشارات

www.manaraa.com

programming (HTML and JavaScript) and Visual Basic for Applications (VBA, a programming language used by Microsoft Access and Excel).

Table 3

Demographics of Students Who Responded to All Surveys

				Prior Experience		
Group	N	Male	Female	N	Male	Female
Experimental	28	21	7	12	10	2
Control	15	10	5	7	4	3

Independence and Persistence (Factor 1)

Questions number 16, 18, 19, 20, 21, 22, 23, and 24 (see Appendix A) related to the Factor 1. This factor was designed to measure a student's ability to work independently and to continue working despite various levels of difficulty encountered with their programming tasks (Ramalingam & Wiedenbeck, 1998).

Table 4 shows the mean and standard deviations for the Factor 1 questions at the pre-, mid- and post-test points for the both groups of students. Each group showed an increase in the mean over time, indicating that, as expected, students felt more confident in performing these tasks as the course progressed.

المنارة للاستشارات

www.manaraa.com

Table 4

Factor 1 Descriptive Statistics

Group	Pre-Test			Mid-Test			Post-Test		
	Mean	SD	N	Mean	SD	N	Mean	SD	N
Experimental	3.88	1.870	28	4.95	1.330	28	5.52	1.028	28
Control	2.66	0.890	15	5.62	0.739	15	5.68	0.913	15
Total	3.46	1.691	43	5.19	1.192	43	5.58	0.981	43

Cronbach's *alpha* for Factor 1 was 0.756. While lower than published validation studies, it meets the criteria established by Gay, Mills and Airasian (2009, p. 162):

> "Standardized achievement and aptitude tests should have high reliability, often higher than 90. On the other hand, personality measures and other non-projective tests do not typically report such high reliabilities."

To determine if there was a significant difference between the self-efficacy in Factor 1 reported by the experimental group when compared to the control group, scores were analyzed using an analysis of covariance (see Table 5, below). Due to a high degree of correlation, the pre-test and mid-test scores were combined and used as the covariate. This allowed results to be analyzed using a single test on the Factor 1 post-test scores.

The Group row of Table 5 shows the analysis by control and experimental groups on the Factor 1 post-test scores using the covariate. There is no significant difference shown between the control and experimental groups (p = 0.753) in self-efficacy related to Factor 1. The low effect size (partial *eta* squared = 0.003) supports this.

www.manaraa.com

Table 5

Factor 1 Tests of Between-Subjects Effects

Source	Sum of Squares	*df*	Mean Square	*F*	*p*	Partial η^2
Corrected Model	12.639[a]	2	6.320	9.091	0.001	0.313
Group	0.070	1	0.070	0.100	0.753	0.003
Error	27.806	40	0.695			

Note: $R^2 = 0.313$ (Adjusted $R^2 = 0.278$)

Testing of Sub-Hypothesis One

These results do not support the rejection of the first sub-hypothesis; instruction in specific non-mathematical problem-solving skills did not lead to significant differences in levels of student self-efficacy pertaining to independence and persistence.

Complex Programming Tasks (Factor 2)

Questions number 7, 10, 11, 12, 13, 14, 15, 17, 27, 28 and 31 (see Appendix A) related to the second factor of Complex Programming Tasks. Factor 2 was designed to measure a student's ability to carry out complex programming tasks such as designing, understanding, changing, and debugging complex programs and reusing code written by others (Ramalingam & Wiedenbeck, 1998).

www.manaraa.com

Table 6

Factor 2 Descriptive Statistics

Group	Pre-Test			Mid-Test			Post-Test		
	Mean	SD	N	Mean	SD	N	Mean	SD	N
Experimental	3.08	1.705	28	4.40	1.415	28	5.33	1.012	28
Control	2.18	0.757	15	4.49	0.673	15	5.10	1.305	15
Total	2.77	1.500	43	4.43	1.200	43	5.25	1.113	43

Table 6 shows the mean and standard deviations for the Factor 2 questions at the pre-test, mid-test and post-test points for the both groups of students. As expected, each group showed an increase in the mean over time, indicating that students felt more confident in performing these tasks as the course progressed.

To determine if there was a significant difference between the self-efficacy in Factor 2 reported by the experimental group when compared to the control group, scores were analyzed using an analysis of covariance (see Table 7). Due to a high degree of correlation, the pre-test and mid-test scores were combined and used as the covariate. Combined with an acceptable Cronbach's *alpha* value of 0.781, this allowed results to be analyzed using a single test on the Factor 2 post-test scores.

المنارة للاستشارات

www.manaraa.com

Table 7

Factor 2 Tests of Between-Subjects Effects

Source	Sum of Squares	*df*	Mean Square	*F*	*p*	Partial η^2
Corrected Model	15.683[a]	2	7.841	8.626	0.001	0.301
Group	0.453	1	0.453	0.498	0.484	0.012
Error	36.363	40	0.909			

Note: $R^2 = 0.301$ (Adjusted $R^2 = 0.266$)

The Group row of Table 7 shows the analysis by control and experimental groups on the Factor 2 post-test scores using the covariate. There is no significant difference shown between the control and experimental groups ($p = 0.484$) in self-efficacy related to Factor 2. The low effect size (partial *eta* squared = 0.012) supports this.

Testing of Sub-Hypothesis Two

These results do not support the rejection of the second sub-hypothesis; instruction in specific non-mathematical problem-solving skills did not lead to significant differences in levels of student self-efficacy pertaining to complex programming tasks.

Self-Regulation (Factor 3)

Questions number 25, 26, 29 and 30 (see Appendix A) related to the third factor of Self-Regulation. This factor was designed to measure a student's ability to control and manage oneself in order to reach a desired end. This includes persistence when disinterested and an ability to work under time pressures (Ramalingam & Wiedenbeck, 1998).

المنارة للاستشارات

www.manaraa.com

Table 8

Factor 3 Descriptive Statistics

Group	Pre-Test			Mid-Test			Post-Test		
	Mean	SD	N	Mean	SD	N	Mean	SD	N
Experimental	4.29	1.765	28	4.71	1.432	28	5.25	1.284	28
Control	2.85	1.012	15	5.05	0.872	15	5.33	1.088	15
Total	3.79	1.682	43	4.83	1.265	43	5.28	1.207	43

Table 8 shows the mean and standard deviations for the Factor 3 questions at the pre-, mid- and post-test points for the both groups of students. Each group showed an increase in the mean over time, indicating that students felt more confident in performing these tasks as the study progressed.

To determine if there was a significant difference between the self-efficacy in Factor 3 reported by the experimental group when compared to the control group, scores were analyzed using an analysis of covariance (see Table 9, below). Due to a high degree of correlation, the pre-test and mid-test scores were combined and used as the covariate. Combined with an acceptable Cronbach's alpha of 0.746, this allowed results to be analyzed using a single test on the Factor 3 post-test scores.

المنارة للاستشارات

www.manaraa.com

Table 9

Factor 3 Tests of Between-Subjects Effects

Source	Sum of Squares	*df*	Mean Square	*F*	*p*	Partial η^2
Corrected Model	24.538[a]	2	12.269	13.404	0.000	0.401
Group	0.210	1	0.210	0.230	0.634	0.006
Error	36.613	40	0.915			

Note: $R^2 = 0.301$ (Adjusted $R^2 = 0.266$)

The Group row of Table 9 shows the analysis by control and experimental groups on the factor 3 post-test scores using the covariate. There is no significant difference shown between the control and experimental groups ($p = 0.634$) in self-efficacy related to Factor 3. The low effect size (partial *eta* squared = 0.006) supports this.

Testing of Sub-Hypothesis Three

These results do not support the rejection of the third sub-hypothesis; instruction in specific non-mathematical problem-solving skills did not lead to significant differences in levels of student self-efficacy related to self-regulation.

Simple Programming Tasks (Factor 4)

Questions number 1, 2, 3, 4, 5, 6, 8, and 9 (see Appendix A) related to the fourth factor of Simple Programming Tasks. This factor was designed to measure a student's ability to perform simple and intermediate programming tasks such as writing simple blocks of code, designing and implementing small to medium sized programs, and simple debugging activities (Ramalingam & Wiedenbeck, 1998).

المنارة للاستشارات

www.manaraa.com

Table 10

Factor 4 Descriptive Statistics

Group	Pre-Test			Mid-Test			Post-Test		
	Mean	SD	N	Mean	SD	N	Mean	SD	N
Experimental	2.75	1.772	28	4.74	1.216	28	5.78	0.895	28
Control	2.43	1.073	15	5.01	0.886	15	5.89	0.838	15
Total	2.64	1.558	43	4.83	1.109	43	5.82	0.867	43

Table 10 shows the mean and standard deviations for the Factor 4 questions at the pre-test, mid-test and post-test points for the both groups of students. Each group showed an increase in the mean over time. As expected, this indicates that students felt more confident in performing these tasks as the course progressed.

Table 11

Factor 4 Tests of Between-Subjects Effects

Source	Sum of Squares	*df*	Mean Square	*F*	*p*	Partial η^2
Corrected Model	9.621[a]	2	4.810	8.766	0.001	0.305
Group	0.002	1	0.002	0.004	0.949	0.000
Error	21.950	40	0.549			

Note: $R^2 = 0.305$ (Adjusted $R^2 = 0.270$)

To determine if there was a significant difference between the self-efficacy in Factor 4 reported by the experimental group when compared to the control group, scores

www.manaraa.com

were analyzed using an analysis of covariance (see Table 11, above). Due to a high degree of correlation, the pre-test and mid-test scores were combined and used as the covariate. Combined with an acceptable Cronbach's alpha of 0.691, this allowed results to be analyzed using a single test on the Factor 4 post-test scores.

The Group row of Table 11 shows the analysis by control and experimental groups on the Factor 4 post-test scores using the covariate. There is no significant difference shown between the control and experimental groups (p = 0.949) in self-efficacy related to Factor 4. The low effect size (partial *eta* squared = 0.000) supports this.

Testing of Sub-Hypothesis Four

These results do not support the rejection of the fourth sub-hypothesis; instruction in specific non-mathematical problem-solving skills did not lead to significant differences in levels of student self-efficacy related to performance of simple programming tasks.

Testing of Hypothesis One

The combination of the data analysis of the four factors above does not support the rejection of the first hypothesis; instruction in specific non-mathematical problem-solving skills did not lead to significantly higher self-efficacy in computer programming tasks.

Achievement

Midterm and final course grade information for each student who completed all three surveys was collected at the end of the term. These scores are the current average of all tests, quizzes, and programming assignments at the middle and end of the term. They

المنارة للاستشارات

www.manaraa.com

were evaluated using an ANCOVA to determine if there was a significant difference in achievement for students who received the experimental treatment.

Table 12

Achievement Descriptive Statistics

Group	Midterm			Final		
	Mean	SD	N	Mean	SD	N
Experimental	72.25	16.816	28	86.89	9.758	28
Control	82.33	24.183	15	79.40	28.256	15
Total	75.77	20.009	43	84.28	18.450	43

Table 12 shows the descriptive statistics for student achievement. The experimental group shows an increase in mean from the middle to the end of the term. The control group midterm score mean was higher than the experimental group midterm mean, however, the control group mean actually decreased between the middle and end of the term.

المنارة للاستشارات
www.manaraa.com

Table 13

Achievement Between-Subjects Effects

Source	Sum of Squares	*df*	Mean Square	*F*	*p*	Partial η^2
Corrected Model	9490.343[a]	2	4745.172	39.491	0.000	0.664
Group	2088.084	1	2088.084	17.378	0.000	0.303
Error	4806.308	40	120.158			

Note: $R^2 = 0.301$ (Adjusted $R^2 = 0.647$)

To determine if there was a significant difference between the achievement of the experimental group when compared to the control group, scores were analyzed using an analysis of covariance (see Table 13, above). The dependent variable was the final grade for the course. The midterm score was used as the covariate.

The Group row of Table 13 shows the analysis by control and experimental groups on achievement using the covariate to control for the initial difference. There is a significant difference shown between the control and experimental groups ($p = 0.000$) in achievement. The effect size (partial *eta* squared = 0.303), however, is moderately small likely demonstrating the disordinal relationship between the pre and post achievement of the control and experimental groups.

Testing of Hypothesis Two

These results support the rejection of the second research hypothesis; there was a significant difference in achievement between the two groups when controlling for earlier scores.

www.manaraa.com

Summary of Results

An analysis of the descriptive and inferential statistics demonstrated that there was no significant difference in self-efficacy when students receive specific non-mathematical problem-solving instruction. There was a significant difference in achievement for the students who received problem-solving instruction. Due to the small number of results collected on all three surveys, there was insufficient data collected to analyze results based on gender or prior experience. These results were not as hypothesized; possible reasons and suggestions for future research will be presented in Chapter 5.

www.manaraa.com

Chapter 5

Conclusions, Implications, Recommendations, and Summary

Conclusions

This study investigated the following research questions:

- What are the primary causes of the declining enrollment of students in computer science?
- What techniques have been implemented to increase student enrollment? What has been the success of these techniques?
- How does specific instruction in non-mathematical problem-solving techniques and critical thinking skills impact the students' computer programming self-efficacy and achievement?

The research hypotheses for the study focused on the last research question. The first hypothesis stated that students who received instruction in non-mathematical problem-solving and critical thinking skills prior to programming instruction would exhibit significantly higher self-efficacy in computer programming tasks than students who did not receive that instruction. There were four sub-hypotheses to hypothesis one as programming self-efficacy was measured through the Computer Programming Self-Efficacy Survey (CPSES), which measures self-efficacy based on four factors. Each factor, therefore, had a sub-hypothesis stating that students who received the problem-solving and critical thinking instruction would have higher self-efficacy in that factor than students who did not receive the instruction. The four factors measured by the CPSES were: students' feelings about their ability to work independently and persist in

www.manaraa.com

the face of hardship, perform complex programming tasks, motivate themselves, and perform simple programming tasks. A second hypothesis stated that students who received non-mathematical problem-solving and critical thinking instruction would exhibit significantly higher achievement than students who did not receive this instruction.

The independent variable in this study was participation in the problem-solving tutorial. There were two dependent variables: achievement (measured by mid-term and final student grades), and programming self-efficacy (measured by the CPSES).

The data revealed no significant difference in self-efficacy between students who received the non-mathematical problem-solving and critical thinking instruction and those who did not. Thus there was a failure to reject the null hypothesis for hypothesis one and its four sub-hypotheses.

The data revealed that students in the experimental group had significantly higher achievement than students in the control group. The null hypothesis for hypothesis two was, therefore, rejected. While this shows statistical significance, due to the effect size and disordinal nature of the data between groups, care has to be taken in its interpretation.

Participation in the study declined over the course of the term. Because self-efficacy was measured over time, only data from students who participated in all three surveys and submitted acknowledgment of tutorial completion in the first week was able to be used. Ultimately, not enough data was available over all three surveys to evaluate the effect of the experimental tutorial specifically on female students' self-efficacy and achievement as compared to their male counterparts.

المنارة للاستشارات
www.manaraa.com

Student Non-Participation

Students in the study took the tutorial as requested, however, they did not complete and turn in the worksheets through the term; this conceivably negatively affected the results of this study. As Kibble (2011) noted, students tend not to participate in online activities that are optional and ungraded. The question then becomes “why?” Drawing from the literature on student motivation, the answers are multitude and focus on both the students and teacher.

Dobson (2008) demonstrated the positive influence of such activities. In a study involving the encouragement of students to read material before it was presented in class, students in an experimental group greatly out-performed their peers. Others (Lei, Bartlett, Gorney, & Herschbach, 2010; Anderson, Teraban, & Sharman, 2003) note that, while evidence does support the efficacy of these types of assignments, there are many reasons that students do not comply:

1. Low student self-confidence.
2. Perceived lack of importance unless assigned as part of the class assignments.
3. Lack of interest in the subject matter.
4. Underestimating the value of the supplemental activity.

At the same time, many instructors are hesitant to require ancillary assignments because of:

1. Fear of poor student evaluations of instructors.
2. The developmental level of students.

المنارة للاستشارات
www.manaraa.com

3. The motivational levels of students and instructors.
4. Individual instructor expectations and beliefs.

Perhaps the single largest factor related to non-use of ancillary material or non-compliance with assignments not directly assigned by the instructor is one of a lack of student motivation. Motivation to become involved in these activities cannot be increased when students do not have the intrinsic desire or perceive extrinsic reward (Deci & Ryan, 1985; Lepper, Greene, & Nisbett, 1973). In this case, faculty involved in this study indicated they asked students to return the worksheets but it is highly likely that students' *state motivation* (i.e. motivation to completion at task at a given point in time) was not of the degree necessary to comply with the teacher's requests (Christophel, 1990; Christophel & Gorham, 1995). It is quite possible that could be attributed to a perceived lack of instructor immediacy; as Rocca (2004) points out, unless an instructor is open with students and explains the reasoning and importance of a given assignment (i.e. establishes immediacy), it is likely that participation will be negatively affected. Suggestions for addressing each of these compliance issues will be addressed in the Recommendations section below.

Implications

Providing students with specific instruction in non-mathematical problem-solving skills appears to have had a positive impact on overall student achievement in an introductory programming course. Achievement has been tied to persistence and self-efficacy in computer science programs (e.g. Arshad, 2009; Baker et al., 2007; Beckwith et al., 2006). When students are able to complete their assignments successfully and

المنارة للاستشارات

www.manaraa.com

maintain a grade with which they are content, they are less likely to be discouraged and choose a different program of study. Building a problem-solving foundation into the introductory courses sets the stage for continued success in future courses (Beaubouef & McDowell, 2008; DeClue, 2008; Goold & Rimmer, 2000).

Student self-efficacy improved in both the control and experimental groups, with no significant difference between the two. Programming is a skill that is best developed through application, thus some increase in self-efficacy due to completion of the course was anticipated. It was anticipated that gaining skill in problem-solving techniques, which would aid in the completion of course assignments, would add to that self-efficacy; however, the data did not support this hypothesis.

Although there was not enough data collected to statistically analyze the effect on self-efficacy and achievement by gender, differences in means were observed between the genders. Females in the experimental group had a higher than average degree of self-efficacy in factors one, two, and four (independence and persistence, complex programming tasks, and simple programming tasks) when compared with males in the experimental group and both genders in the control group. Females in the experimental group had lower than average self-efficacy scores on factor three (self-motivation) when compared to males in both groups, but did score higher than their female peers in the control group.

Recommendations

The problem-solving tutorial used in this study was provided as an online resource to be taken by participating students on their own time during the first week of

المنارة للاستشارات
www.manaraa.com

the course. The additional worksheet which assisted students in applying the problem-solving skills to their assignments was also an online resource to be completed in the student's own time. It became apparent that there were several issues that affected the validity of these results. Because of this, the following suggestions for future research should be considered:

1. It is recommended that future research incorporate both of these resources into the requirements for the course, providing time for the tutorial material to be covered in class and requiring the worksheet as part of the completed assignment. This may ensure that students in the experimental group received the full treatment in a more consistent and verifiable format. Though all experimental students indicated their completion of the tutorial in the first week, the failure of these students to complete the problem-solving worksheets with each assignment may account for the lack of a significant difference in self-efficacy between the groups.
2. This study was conducted in the spring term. Traditionally there is larger enrollment in introductory programming classes during the fall term. Conducting the study during this term would provide a larger population for study that could result in sufficient final data to more thoroughly investigate the problem-solving instruction's impact by gender.
3. As conducted, students were required to review the tutorial only once. Consideration should be given to requiring it again further into the term.

المنارة للاستشارات

www.manaraa.com

4. Self-efficacy beliefs about computers can be formed as early as the end of elementary school (Liu et al., 2006; Tang et al., 2008; Pintrich & Schunk, 1996). Typically grade school students are not exposed to programming; however, often high school students have the opportunity to participate in programming as an elective course (Mayall, 2008; Nauta & Epperson, 2003). Replicating this study at the high school level could provide interesting insight into what keeps students from choosing computer science when they enter college.
5. The disordinal interaction of the achievement scores could better be understood by interviews with faculty and students. Designing this study in a mixed-methods format would perhaps lend itself to a better understanding and interpretation of the results (Gay, Mills & Airasian, 2006).
6. While the reliability of the self-efficacy measurement instrument was acceptable, due to the dated establishment of the psychometrics of the instrument, consideration should be given to attempting to locate and use a more recent instrument.
7. Due to the subjective nature of grading, it is possible that differences in achievement scores could be reflective of the interpretation of given test answers by a particular instructor. Given that, thought should be given to obtaining or developing a standardized instrument to be used by all instructors.

المنارة للاستشارات
www.manaraa.com

8. Grades were based on a combination of examinations, quizzes, and programming assignments. An extremely low or high grade on a given point of measurement could radically affect the grades of students in a given class. Care should be taken to ensure that the same number of evaluations occurred for all students.
9. Grades were based on a combination of examinations, quizzes, and programming assignments. Consideration should be given to comparing the data separately for each of these types of evaluation.
10. Achievement data was collected at three points during the term – pre-term, mid-term, and final. In order to investigate the effect of immediacy in the experimental group, consideration should be given to measure achievement at lesser intervals.
11. In order to truly test the effect of the intervention on achievement and efficacy, a longitudinal perspective should be considered. Administering achievement tests each week would allow for examining fluctuations in achievement.
12. Due to the self-selection of students into the classes involved in the study, the results may be biased. The study should be replicated in programming classes where attendance is required.
13. The elective nature of this course may have affected the results. Consideration of conducting the study in a required course is called for.

المنارة للاستشارات
www.manaraa.com

Summary

The purpose of this study was to investigate the impact of non-mathematical problem-solving instruction on the self-efficacy and achievement of college-level introductory programming students. It was anticipated that this instruction would increase student self-efficacy and achievement.

Five sections of an introductory programming course at a community college participated in this study. The students who agreed to participate in the experimental group took an online tutorial in non-mathematical problem-solving skills. The control group received the standard introductory programming course. The data that was collected and analyzed showed no significant difference in self-efficacy between the two groups. Failure to administer the full experimental treatment may have contributed to this lack of significant difference. The experimental group did have significantly higher achievement than the control.

Insufficient data was collected to analyze the effect of the tutorial by gender. A positive difference in means was observed in three of the four self-efficacy factors measured between females in the experimental group when compared with their peers (both control and experimental).

Ultimately, the results of this study suggest that specific non-mathematical problem-solving instruction may have a positive effect on student achievement and may provide a step toward increasing student programming self-efficacy. These results, however, are tenuous at best. Further research is called for in this arena.

المنارة للاستشارات

www.manaraa.com

Appendix A – Computer Programming Self-Efficacy Scale

Rate your confidence in doing the following programming related tasks using a scale of 1 (not at all confident) to 7 (absolutely confident). If a specific term or task is totally unfamiliar to you, please mark 1.

1. Write syntactically correct statements in Java.	1	2	3	4	5	6	7
2. Understand the Java language structure and usage of the reserved words.	1	2	3	4	5	6	7
3. Write logically correct blocks of code in Java.	1	2	3	4	5	6	7
4. Write a Java program that displays a greeting message.	1	2	3	4	5	6	7
5. Write a program in Java that computes the average of three numbers.	1	2	3	4	5	6	7
6. Use built-in functions that are available in various Java libraries.	1	2	3	4	5	6	7
7. Build my own built-in function or library in Java.	1	2	3	4	5	6	7
8. Write a small program given a small problem that is familiar to me.	1	2	3	4	5	6	7
9. Write a reasonably sized program that can solve a problem that is only vaguely familiar to me.	1	2	3	4	5	6	7
10. Write a long and complex Java program to solve any given problem as long as the specifications are clearly defined.	1	2	3	4	5	6	7
11. Organize and design my program in a modular manner.	1	2	3	4	5	6	7
12. Understand the object-oriented paradigm.	1	2	3	4	5	6	7
13. Identify the objects in the problem domain and declare, define, and use them.	1	2	3	4	5	6	7
14. Make use of a pre-written function, given a clearly labeled declaration of the function.	1	2	3	4	5	6	7
15. Make use of a class that is already defined given a clearly labeled declaration of the class.	1	2	3	4	5	6	7
16. Debug (correct all errors) a long and	1	2	3	4	5	6	7

المنارة للاستشارات
www.manaraa.com

complex program that I have written and make it work.							
17. Comprehend a long, complex program.	1	2	3	4	5	6	7
18. Complete a programming project if someone showed me how to solve the problem first.	1	2	3	4	5	6	7
19. Complete a programming project if I had only the language reference manual for help.	1	2	3	4	5	6	7
20. Complete a programming project if I could call someone for help if I got stuck.	1	2	3	4	5	6	7
21. Complete a programming project once someone else helped me get started.	1	2	3	4	5	6	7
22. Complete a programming project if I had a lot of time to complete the program.	1	2	3	4	5	6	7
23. Complete a programming project if I had just the built-in help facility for assistance.	1	2	3	4	5	6	7
24. Find ways of overcoming the problem if I got stuck at a point while working on a programming project.	1	2	3	4	5	6	7
25. Come up with a suitable strategy for a given programming project in a short time.	1	2	3	4	5	6	7
26. Manage my time efficiently if I had a pressing deadline on a programming project.	1	2	3	4	5	6	7
27. Mentally trace through the execution of a long, complex program given to me.	1	2	3	4	5	6	7
28. Rewrite lengthy confusing portions of code to be more readable and clear.	1	2	3	4	5	6	7
29. Find a way to concentrate on my program, even when there were many distractions around me.	1	2	3	4	5	6	7
30. Find ways of motivating myself to program, even if the problem area was of no interest to me.	1	2	3	4	5	6	7
31. Write a program that someone else could comprehend and add features to at a later date.	1	2	3	4	5	6	7

المنارة للاستشارات

www.manaraa.com

Scale taken from:

Ramalingam, V. & Wiedenbeck, S. (1998). Development and validation of scores on a computer programming self-efficacy scale and group analysis of novice programmer self-efficacy. *Journal of Educational Computing Research, 19*(4), 367 – 381.

www.manaraa.com

Appendix B – Supplemental Materials for Experimental Classes

Tutorial Content and Exercises

Problem Solving

- Problem solving is the process used to find answers in all kinds of situations.
  - From making sure you got correct change to figuring out how to get all your friends to Prom for the least amount of money.
- Problem solving skills are also critical for designing solutions to word problems like we see in math and computer science classes.

Problem Solving Pitfalls

- Reading inaccuracy
  - Since word problems generally occur as paragraphs, it's tempting to try and skim through to find the pertinent information and be done. Unfortunately this often means:
    - that we choose the wrong information as relevant,
    - we fail to truly understand what the problem really is,
    - we miss key information that is crucial to a solution, or
    - we misunderstand what the solution should look like.

www.manaraa.com

Problem Solving Pitfalls, con’t

- Thinking inaccuracy
  - This pitfall follows close on the heels of Reading Inaccuracy. In fact, one often causes the other. Thinking inaccuracy generally means:
    - that we were inconsistent in how we interpreted words or performed actions,
    - we skipped a final check of our solution,
    - we went with our "first guess" for the solution rather than considering all the tools at our disposal, or
    - we made up our mind about how to do something as we were reading rather than waiting until we had all the facts.

Problem Solving Pitfalls, con’t

- Weak analysis
  - Weak analysis is one of the more critical pitfalls since slowing down usually isn't enough to completely eliminate it. When we have weak analysis, it generally means:
    - that we didn't break the problem into smaller, more solvable parts,
    - we didn't consider all our previous problem solving experiences when trying to make sense of the matter at hand,
    - we skipped over ideas or words that were unfamiliar rather than researching until we had a good understanding, or
    - we didn't use drawings or notes to help us formulate our solutions.

المنارة للاستشارات
www.manaraa.com

Problem Solving Pitfalls, con’t

- Giving up too easily
  - Just like in life, problem solving requires perseverance, even when the problem is difficult and takes longer than expected! Often we give up when:
    - we have low confidence in our ability to actually solve the problem and decide to not "waste our time" when "we're not going to succeed anyway",
    - we try the first thing that comes to mind and either hit a roadblock or get some kind of solution and we figure something is better than nothing,
    - we go through the motions without really thinking about what we're doing, or
    - we get frustrated half-way through and so just jump to a conclusion rather than seeing the thought process through to the end.
  - When problem solving it's important that we keep a positive attitude and not let self-doubt creep in and dissuade us from putting our best effort into a solution.
  - Solving problems takes time!

Problem Solving Pitfalls, con’t

- Not thinking aloud
  - **Basically, this involves vocalizing everything you think as you solve a problem.**
- Over time, you will learn to think aloud in your head (silently!)
- In the mean time, working with a listening partner can help you practice your thinking aloud.
- Let’s look at the think aloud example.

المنارة للاستشارات
www.manaraa.com

A Think-Aloud Example

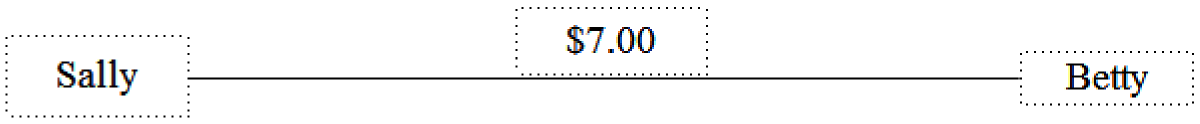
The example problem is a typical math word problem. Don't concentrate on the actual solution, but instead consider the method used to arrive at that solution. Listen (or read) to the entire thought process as its spoken, follow along. Once you've listened through the problem, start at the top and read through it without the sound, as if you were solving the problem using the think-aloud method.

Example Problem:

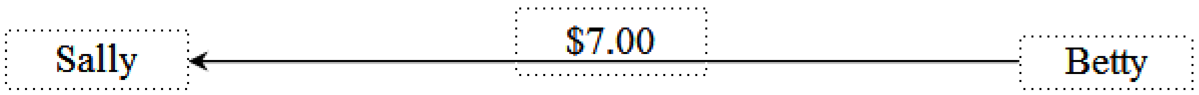
Sally loaned $7.00 to Betty. But Sally borrowed $15.00 from Estella and $32.00 from Joan. Moreover, Joan owes $3.00 to Estella and $7.00 to Betty. One day the girls got together at Betty's house to straighten out their accounts. Which girl left with $18.00 more than she came with?

The solution:

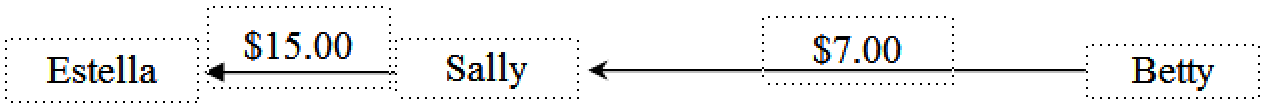
"First I read the problem out loud. Ok, I think I need a diagram of some sort to show who owes money to whom to try and keep it straight, so I'll start with the first sentence and draw Sally and Betty and the $7.00."



"Ok, that looks good, but I can't tell who owes the money, so I should probably change the line into an arrow showing that Betty owes the money to Sally, which makes the drawing look like this:"

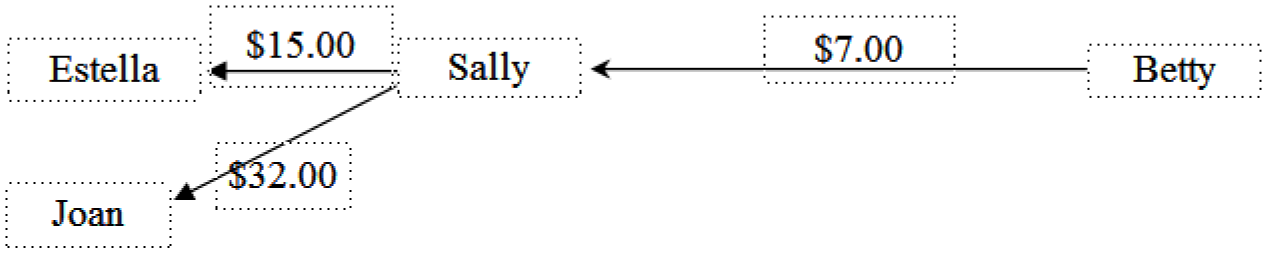


"Alright, now, what's next? The second sentence says that Sally borrowed $15.00 from Estella - I'll stop there and add it to my diagram, so I need a line from Sally off to a new girl named Estella, and since Sally borrowed from her, the arrow needs to point at Estella."

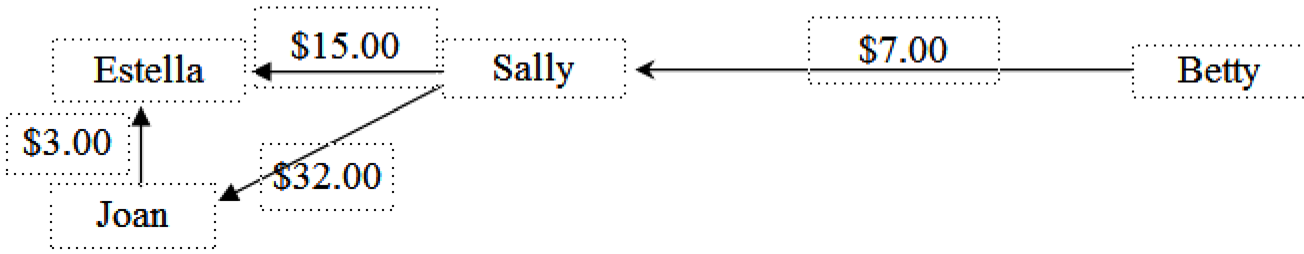


www.manaraa.com

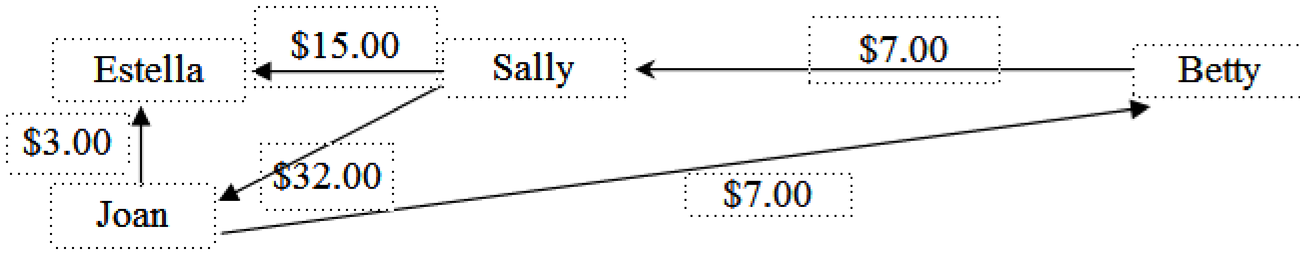
"Ok, the rest of the line says 'and $32.00 from Joan.' So, Sally borrowed $32.00 from Joan in addition to the $15.00 from Estella. So I need to add another arrow from Sally, pointing to Joan with a label of $32.00. Gosh, Sally borrows a lot of money from people. Let's see, my drawing should now look like this:"



"Great. What's next? The third sentence says that Joan owes $3.00 to Estella - I'll stop there and add it to my drawing, the arrow points at Estella since Joan owes the money."



"And then it says 'and $7.00 to Betty' - so Joan owes Betty $7.00. Let me add that to the drawing, and the arrow would point at Betty because Joan owes the money."



"Wow, ok, what's next? Oh, good, they're getting together at Betty's house to straighten out their accounts. Do I care about any of that information? I can't see what it would matter that they were at Betty's house, unless she charged them admission, but it doesn't say anything about that. So I'll just keep reading, Which girl left with $18.00 more

www.manaraa.com

than she came with? Ok, well, let's look at the diagram; I'll start with Joan since she's on the bottom."

"Joan has to pay $3.00 to Estella and $7.00 to Betty, but she's getting $32.00 from Sally. So at the end of the day, she pays out $10.00, because that's $3 plus $7, and gets $32.00, which means Joan leaves with $22.00."

"Betty is paying $7.00 to Sally and getting $7.00 from Joan, so she is breaking even and leaving with the same $7.00 she came with."

"Sally is getting $7.00, but paying out $15 to Estella and $32.00 to Joan. So Sally pays a total of $15 plus $32, which is $47.00. So Sally leaves with $40.00 less than what she brought."

"Finally, Estella. So far no one else has $18.00 more, so I'm pretty sure the answer is Estella, but I should check the numbers just to be sure I didn't make a mistake somewhere else. Estella didn't owe anyone money, so she didn't bring anything with her. But she does get $3.00 from Joan and $15.00 from Sally and $3.00 plus $15.00 is $18.00. So I'm right, Estella leaves with $18.00 more than she brought."

المنارة للاستشارات
www.manaraa.com

Identifying Relevant Information

- Relevant information is critical to the solution of the problem.
- All relevant information must be used in the problem solution.
- Irrelevant information is not used in the problem solution.
- Irrelevant information can cause you to make mistakes when designing your solution.

Identifying Relevant Information

- The key is to ask yourself, "Does this information help me solve the problem?"
  - If the answer is yes, then it's relevant.
  - If the answer is no, then it's irrelevant and you can disregard it.
  - If the answer is, I'm not sure, then make a note of it and come back later as you do your design.
- Let's talk through an example, using the think aloud method.

المنارة للاستشارات

www.manaraa.com

Example Problem:

Tropical depressions, tropical storms and hurricanes are classified by their wind speeds. A tropical depression has wind speeds that range from 0 to 38 miles per hour. The wind speeds of a tropical storm range from 39 to 73 miles per hour. Storms with winds greater than 73 miles per hour constitute hurricanes, which are classified into categories that range from 1 to 5 depending on the wind speed of the storm. Wind speeds of a category 1 storm range from 74 to 95 miles per hour. Category 2 wind speeds range from 96 to 110 miles per hour. Category 3 wind speeds range from 111 to 130 miles per hour. Category 4 wind speeds range from 131 to 155 miles per hour, and wind speeds over 155 miles per hour define a category 5 hurricane. Hurricane Katrina was a category 4 storm when it made landfall with wind speeds of 140 miles per hour. Hurricane Andrew was a category 5 at landfall, with wind speeds of 165 miles per hour. Both storms caused excessive damage to the surrounding areas. Write a computer program that indicates the type and category of a storm when given the current wind speed.

The Solution:

First, I read the problem out loud. While I do this, I am thinking to myself "What is the problem I'm solving?" I want to determine what it is I'm trying to do because that will help me distinguish between relevant and irrelevant information. I think it's in the last line; I want to write a computer program that indicates the type and category of a storm when I know the current wind speed. So I'm looking for both the type of storm and the category of that storm. What do they mean by type of storm, I wonder? Let me look back at the problem statement - ok, there in the first line they talk about tropical depressions, tropical storms, and hurricanes. So I'm to decide if it's one of those. So, then what about category? Hurricanes are the only storms with categories, it looks like, and so if the storm is a hurricane, then I need to see what category of hurricane it is.

Ok, so now I know what I'm trying to do. The next thing I need to do is decide what information will help me do that and what information will not. I think the best way to do that is to make two lists - I'll read through the problem again and on the left side I'll list all the relevant information and on the right side I'll list the irrelevant information. So I'll start with a chart like this:

المنارة للاستشارات

www.manaraa.com

Relevant	Irrelevant

The best way to approach a problem is in small steps, so I'll take each sentence in turn and identify the information and classify it as relevant or irrelevant. Starting with the first sentence: "Tropical depressions, tropical storms and hurricanes are classified by their wind speeds." Ok, now the question I need to ask is, *does this help me solve the problem?* It gives me some generic background information and helps me understand the point of the problem somewhat, but no, it really does very little to help me solve the problem. So I'm going to categorize this as irrelevant. Adding that to the chart gives me the following:

Relevant	Irrelevant
	Tropical depressions, tropical storms and hurricanes are classified by their wind speeds.

المنارة للاستشارات

www.manaraa.com

Great - moving on to sentence two: "A tropical depression has wind speeds that range from 0 to 38 miles per hour." Ok, again I ask if this helps me solve the problem. Yes! Since I need to classify a given wind speed, I need to know how I would classify it as a tropical depression and this sentence gives me just that information. So I will add it to the relevant side of my chart.

Relevant	Irrelevant
Tropical depression = 0 – 38 mph	Tropical depressions, tropical storms and hurricanes are classified by their wind speeds.

Now we move on to the third sentence: "The wind speeds of a tropical storm range from 39 to 73 miles per hour." This is another very relevant piece of information because it helps us identify tropical storms! So we add it to the relevant side of our chart.

المنارة للاستشارات
www.manaraa.com

Relevant	Irrelevant
Tropical depression = 0 – 38 mph Tropical storm = 39 – 73 mph	Tropical depressions, tropical storms and hurricanes are classified by their wind speeds.

On to the fourth sentence: "Storms with winds greater than 73 miles per hour constitute hurricanes, which are classified into categories that range from 1 to 5 depending on the wind speed of the storm." This is relevant since it tells us some good information about hurricanes, that their wind speeds are greater than 73 miles per hour and that they have 5 categories. But I'm not sure if it will help me solve the problem, I need more information than is really given here. I'm going to wait to assign this sentence until I'm finished and see if I still can get something useful from it when I've gone through the rest of the problem.

So let's move on to the fifth sentence for right now: "Wind speeds of a category 1 storm range from 74 to 95 miles per hour." Ok, this is more like what I need - it gives me very good detail on what a category 1 hurricane is. So I'll add this to the relevant side of the chart:

المنارة للاستشارات

www.manaraa.com

Relevant	Irrelevant
Tropical depression = 0 – 38 mph	Tropical depressions, tropical storms and hurricanes are classified by their wind speeds.
Tropical storm = 39 – 73 mph	
Cat 1 hurricane = 74 – 95 mph	

Let's look at the sixth sentence: "Category 2 wind speeds range from 96 to 110 miles per hour." Again, very relevant since it gives us a clear definition of a category 2 hurricane. I'll add it to the chart on the relevant side:

Relevant	Irrelevant
Tropical depression = 0 – 38 mph	Tropical depressions, tropical storms and hurricanes are classified by their wind speeds.
Tropical storm = 39 – 73 mph	
Cat 1 hurricane = 74 – 95 mph	
Cat 2 hurricane = 96 – 110 mph	

Sentence seven says: "Category 3 wind speeds range from 111 to 130 miles per hour." Great! Add that definition of a category 3 hurricane to the relevant side of the chart.

المنارة للاستشارات

www.manaraa.com

Relevant	Irrelevant
Tropical depression = 0 – 38 mph	Tropical depressions, tropical storms and hurricanes are classified by their wind speeds.
Tropical storm = 39 – 73 mph	
Cat 1 hurricane = 74 – 95 mph	
Cat 2 hurricane = 96 – 110 mph	
Cat 3 hurricane = 111 – 130 mph	

And on to sentence eight we go: "Category 4 wind speeds range from 131 to 155 miles per hour, and wind speeds over 155 miles per hour define a category 5 hurricane." Looking at this sentence we can see that it's not only relevant, but it gives us two definitions; one for a category 4 hurricane and one for a category 5 hurricane. Let's add both of those to our chart:

المنارة للاستشارات

www.manaraa.com

Relevant	Irrelevant
Tropical depression = 0 – 38 mph	Tropical depressions, tropical storms and hurricanes are classified by their wind speeds.
Tropical storm = 39 – 73 mph	
Cat 1 hurricane = 74 – 95 mph	
Cat 2 hurricane = 96 – 110 mph	
Cat 3 hurricane = 111 – 130 mph	
Cat 4 hurricane = 131 – 155 mph	
Cat 5 hurricane = 155 mph and up	

Ok. Now, let's move on to the last four sentences and try to take them in one swoop since I think you're probably getting the idea: "Hurricane Katrina was a category 4 storm when it made landfall with wind speeds of 140 miles per hour. Hurricane Andrew was a category 5 at landfall, with wind speeds of 165 miles per hour. Both storms caused excessive damage to the surrounding areas. Write a computer program that indicates the type and category of a storm when given the current wind speed."

Reading those sentences, ask yourself if knowing about hurricane Katrina or Andrew helps you solve the problem. (Remember that the problem is to take a given wind speed and tell what kind of storm it is.) I can't think of how it would - so both of those sentences and the third about the damage caused by both storms, while interesting, do nothing for the problem solution. So we'll classify them as irrelevant. The last sentence tells us what the problem is that we're solving - so it's definitely relevant to the solution! If we add these final pieces of information in, we get our final chart:

المنارة للاستشارات

www.manaraa.com

Relevant	Irrelevant
Tropical depression = 0 – 38 mph	Tropical depressions, tropical storms and hurricanes are classified by their wind speeds.
Tropical storm = 39 – 73 mph	Hurricane Katrina was a cat 4 storm at landfall.
Cat 1 hurricane = 74 – 95 mph	Andrew was a cat 5 storm at landfall.
Cat 2 hurricane = 96 – 110 mph	
Cat 3 hurricane = 111 – 130 mph	
Cat 4 hurricane = 131 – 155 mph	
Cat 5 hurricane = 155 mph and up	
Program should classify storm based on wind speed.	

That's it! We've classified all the information given to us in the problem statement and we're well on our way to solving this problem.

المنارة للاستشارات

www.manaraa.com

Fact Categories

- Relevant information in the problem statement can be classified as three types:
  - Input
  - Output
  - Processes

Inputs

- You can identify what the inputs in a problem statement are by first figuring out what the problem is that you're solving and then asking yourself, "What information do I need in order to start solving the problem?"
- Can be entered by the user
- Can be information used as default settings

المنارة للاستشارات

www.manaraa.com

Outputs

- Outputs are the desired pieces of information that you get from the program. You can usually identify outputs by determining what the problem is that you're solving. If you're not clear, try asking yourself, "What am I trying to do with this program?“
- Can be information that is displayed.
- Generally outputs are the answer to the questions posed in the problem statement.

Processes

- Processes are how we transform input into output. In computer programs, the process portion is generally responsible for the majority of our programming time and effort, so it's critical that we understand exactly what we're doing with our inputs and outputs. If we get the process wrong, it's the same as if we asked someone to build a deck on the back of our house but instead they built us a tree house! The inputs were the same (wood and nails, etc.) but the output is not at all what we wanted.
- Unfortunately, processes are usually the least defined part of the problem statement.

المنارة للاستشارات

www.manaraa.com

Example Problem:

Tropical depressions, tropical storms and hurricanes are classified by their wind speeds. A tropical depression has wind speeds that range from 0 to 38 miles per hour. The wind speeds of a tropical storm range from 39 to 73 miles per hour. Storms with winds greater than 73 miles per hour constitute hurricanes, which are classified into categories that range from 1 to 5 depending on the wind speed of the storm. Wind speeds of a category 1 storm range from 74 to 95 miles per hour. Category 2 wind speeds range from 96 to 110 miles per hour. Category 3 wind speeds range from 111 to 130 miles per hour. Category 4 wind speeds range from 131 to 155 miles per hour, and wind speeds over 155 miles per hour define a category 5 hurricane. Hurricane Katrina was a category 4 storm when it made landfall with wind speeds of 140 miles per hour. Hurricane Andrew was a category 5 at landfall, with wind speeds of 165 miles per hour. Both storms caused excessive damage to the surrounding areas. Write a computer program that indicates the type and category of a storm when given the current wind speed.

The Solution:

I already know (from completing the example in lesson two) that I'm looking for both the type of storm and the category of that storm if it's a hurricane. I also already know that the following information is relevant to solving the problem (and that anything not in this list is irrelevant):

- Tropical Depression wind speed = 0 to 38 mph
- Tropical Storm wind speed = 39 - 73 mph
- Category 1 Hurricane wind speed = 74 - 95 mph
- Category 2 Hurricane wind speed = 96 - 110 mph
- Category 3 Hurricane wind speed = 111 - 130 mph
- Category 4 Hurricane wind speed = 131 - 155 mph
- Category 5 Hurricane wind speed = 156mph and up

Now I want to look for inputs and outputs. I'm going to start with inputs and read the problem out loud again looking for any key phrases that might indicate an input. There, in the last line I see the word "given", does that signify an input? Let's read the sentence again, "Write a computer program that indicates the type and category of a storm when given the current wind speed." So if we're given the current wind speed, then we know that the user is going to give us a wind speed - so our input is current wind speed. Nothing else in the problem statement appears to be an input, so let's move on to outputs.

When I read the last sentence just then I noted that there was another key phrase in addition to "given" - that key phrase is "Write a computer program that..." This is a common phrase to indicate the output and that matches with what we already know is the purpose of our problem from lesson two. So based on lesson two and another look at that statement, I know that my output is going to be the type of storm and the category. But I should probably be a little more specific than "type of storm", I know the types are: Tropical Depression, Tropical Storm, and Hurricane. And I know that if the type of storm is Hurricane then it will have a category ranging from 1 to 5. So my output will be one of the following statements:

- Tropical Depression,
- Tropical Storm,
- Category 1 Hurricane,
- Category 2 Hurricane,
- Category 3 Hurricane,

للاستشارات المنارة

www.manaraa.com

program that indicates the type and category of a storm when given the current wind speed." So if we're given the current wind speed, then we know that the user is going to give us a wind speed - so our input is current wind speed. Nothing else in the problem statement appears to be an input, so let's move on to outputs.

When I read the last sentence just then I noted that there was another key phrase in addition to "given" - that key phrase is "Write a computer program that..." This is a common phrase to indicate the output and that matches with what we already know is the purpose of our problem from lesson two. So based on lesson two and another look at that statement, I know that my output is going to be the type of storm and the category. But I should probably be a little more specific than "type of storm", I know the types are: Tropical Depression, Tropical Storm, and Hurricane. And I know that if the type of storm is Hurricane then it will have a category ranging from 1 to 5. So my output will be one of the following statements:

- Tropical Depression,
- Tropical Storm,
- Category 1 Hurricane,
- Category 2 Hurricane,
- Category 3 Hurricane,
- Category 4 Hurricane, or
- Category 5 Hurricane.

The last thing we need to do for our solution is to look at processes. I know that the processing is going to turn my input, which is a wind speed, into my output, which is one of the statements I just listed. So how do I get from a wind speed to that output? I need to consider what other information I have in the problem statement that I decided was relevant.

Looking back at the relevant information, I can see that so far I really haven't used any of it except the storm types. So it's likely that the information there will help me design my process. If I look at the relevant information and keep in mind that my input is going to be a wind speed, I can start to see that my process is going to have to compare that given wind speed (my input) to the speeds in the range for each storm type. When I find the range that contains my speed, I'm going to output the storm type and, if it's a hurricane, the category for that input wind speed.

Let's try an example and talk through it. Say that our input is a wind speed of 122 miles per hour. If I talk through the process my program needs to follow, it would go something like this:

"Is my wind speed between zero and 38 miles per hour? No. Ok, it's not a Tropical Depression. Is my wind speed between 39 miles per hour and 73 miles per hour? No. So it's not a Tropical Storm. Is my wind speed between 74 and 95 miles per hour? No. So it's not a Category 1 Hurricane. Is my wind speed between 96 and 110 miles per hour?

المنارة للاستشارات

www.manaraa.com

No. So it isn't a Category 2 Hurricane. Is my wind speed between 110 and 130 miles per hour? Yes! 122 miles per hour is between 110 and 130 miles per hour, so this storm is a Category 3 Hurricane."

Talking through an example is a very good way to make sure you have a handle on what exactly your process is going to need to be. For right now, that's good enough. As you get more adept at writing computer programs, you'll get a handle on the specifics of the computer programming language you're working with and be able to translate the process from words into code.

المنارة للاستشارات
www.manaraa.com

Worksheet

For each programming assignment, use this sheet to help you plan your solution prior to beginning programming. Be sure to think-aloud (or think-aloud in your head) and make sure you work through the problem-solving steps you've been taught!

Problem
In your own words, write what you're trying to do.

Relevant Information
List all relevant information from the problem statement that you will need to use in your solution.

Inputs
List the inputs (include things like starting positions, number of beepers, etc.) you'll need for this problem.

Outputs
What is the desired output of the problem?

Processes
What do you have to do to get from your inputs to your outputs? Be specific!

المنارة للاستشارات
www.manaraa.com

Reference List

Adams, J. C. (2007). Alice, middle schoolers & the imaginary worlds camps. *Proceedings of the ACM SIGCSE Technical Symposium on Computer Science Education '07,* 307 – 311.

Adya, M.P. (2008). Work alienation among IT workers: A cross-cultural gender comparison. *Proceedings of the ACM SIGMIS-CPR 2008 Conference,* 66 – 69.

Al-Bow, M., Austin, D., Edgington, J., Fajardo, R., Fishburn, J., Lara, C., et al. (2008). *Proceedings of the 2008 ACM SIGGRAPH symposium on Video games,* 55 – 59.

Ali, S. (2005). Effective teaching pedagogies for undergraduate computer science. *Mathematics and Computer Science Education, 39*(3), 243 – 257.

Allan, V. & Kolesar, M. (1997). Teaching computer science: A problem solving approach that works. *ACM SIGCUE Outlook*, *25*(1), 2 – 10.

Anderson, E., Teraban, R., & Sharman, M. (2003). Student usage of supplementary materials. *Proceedings of the International Conference on Engineering Education, Valencia, Spain.*

Arshad, N. (2009). Teaching programming and problem solving to CS2 students using think-alouds. *Proceedings of the ACM SIGCSE Technical Symposium on Computer Science Education '09,* 372 – 376.

Baker, D., Krause, S., Yasar, S., Roberts, C., & Robinson-Kurpius, S. (2007). An intervention to address gender issues in a course on design, engineering, and technology for science educators. *Journal of Engineering Education, 96*(3), 213 – 226.

Balch, T., Summet, J., Blank, D., Kumar, D., Guzdial, M., O'Hara, K. et al. (2008). Designing personal robots for education: Hardware, software, and curriculum. *Pervasive Computing, 7*(2), 5 – 9.

Ballou, D.J. & Huguenard, B.R. (2008). The impact of students' perceived computer experience on behavior and performance in an introductory information systems course. *Journal of Information Systems Education, 19*(1), 87 – 97.

Bandura, A. (1994). Self-efficacy. In V.S. Ramachaudran (Ed.), *Encyclopedia of human behavior* (Vol. 4, pp. 71 – 81). New York: Academic Press.

المنارة للاستشارات
www.manaraa.com

Barker, L., McDowell, C. & Kalahar, K. (2009). Exploring factors that influence computer science introductory course students to persist. *Proceedings of the ACM SIGCSE Technical Symposium on Computer Science Education '09,* 153 – 157.

Beaubouef, T., Lucas, R., & Howatt, J. (2001). The UNLOCK system: Enhancing problem solving skills in CS-1 students. *ACM SIGCSE Bulletin, 33*(2), 43 – 46.

Beaubouef, T. & McDowell, P. (2008). Computer science: Student myths and misconceptions. *Journal of Computing Sciences in Colleges, 23*(6), 43 – 48.

Beck, J. (2007). Forming a women's computer science support group. *Proceedings of the ACM SIGCSE Technical Symposium on Computer Science Education '07,* 400 – 404.

Beckwith, L., Kissinger, C., Burnett, M., Wiedenbeck, S., Lawrance, J., Blackwell, A., et al. (2006). Tinkering and gender in end-user programmers' debugging. *Proceedings of the ACM CHI 2006 Conference,* 231 – 240.

Berkelaar, B.L., Kisselburgh, L.G., & Buzzanell, P.M. (2008). Locating and disseminating effective messages: Enhancing gender representation in computing majors and careers. *Proceedings of the SIGMIS-CPR 2008 Conference,* 106 – 108.

Biggers, M., Brauer, A., & Yilmaz, T. (2008). Student perceptions of computer science: A retention study comparing graduating seniors vs. CS leavers. *Proceedings of the ACM SIGCSE technical Symposium on Computer Science Education '08,* 402 – 406.

Brown, C., Garavalia, L.S., Fritts, M.L.H., & Olson, E.A. (2006). Computer science majors: Sex role orientation, academic achievement, and social cognitive factors. *The Career Development Quarterly, 54*(4)*,* 331 – 345.

Bureau of Labor Statistics, U.S. Department of Labor. (2007). Occupational Outlook Handbook, 2008-09 Edition, Computer Scientists and Database Administrators. Retrieved September 12, 2008, from http://www.bls.gov/oco/ocos042.htm

Cady, D. & Terrell, S. (2007). The effect of the integration of computing technology in a science curriculum on female students' self-efficacy attitudes. *Journal of Educational Technology Systems, 36*(3), 277 – 286.

Camp, T. (1997). The incredible shrinking pipeline. *Communications of the ACM, 40*(10), 103 – 110.

المنارة للاستشارات

www.manaraa.com

Carr, T., Cox, G., Eden, A., & Hanslo, M. (2004). From peripheral to full participation in a blended trade bargaining simulation. *British Journal of Educational Technology, 35*(2), 197 – 211.

Cassidy, S. & Eachus, P. (2002). Developing the computer user self-efficacy (CUSE) scale: Investigating the relationship between computer self-efficacy, gender and experience with computers. *Journal of Educational Computing Research, 26*(2), 133 – 153.

Cho, M. (1995). Turning Point for Korean Computer Educators: Introducing LogoWriter as a Thinking Tool. *Proceedings of the National Educational Computing Conference: Emerging Technologies, Lifelong Learning, NECC '95*, 237 – 242.

Christophel, D.M. (1990). The relationships among teacher immediacy, behaviors, student motivation, and learning. *Communication Education, 39,* 323 – 340.

Christophel, D.M. & Gorham, J. (1995). A test-retest analysis of student motivation, teacher immediacy, and perceived sources of motivation and demotivation in college classes. *Communication Education, 44,* 292 – 306.

Cohoon, J., Wu, Z., & Chao, J. (2009). Sexism: Toxic to women's persistence in CSE doctoral programs. *Proceedings of the ACM SIGCSE Technical Symposium on Computer Science Education '09,* 158 – 162.

Cohoon, J., Wu, Z., & Luo, L. (2008). Will they stay or will they go? *Proceedings of the ACM SIGCSE technical symposium on Computer Science Education '08,* 397 – 401.

Colley, A., Henry, O., Holmes, S., & James, L. (1996). Perceptions of ability to program or to use a word processor. *Computers in Human Behavior, 12*(3), 329 - 337.

Courte, J. & Bishop-Clarke, C. (2009). Do students differentiate between computing disciplines? *Proceedings of the 40th ACM Technical Symposium on Computer Science Education, 29* – 33.

Daigle, R., Doran, M., & Pardue, J. (1996). Integrating collaborative problem solving throughout the curriculum. *Proceedings of the twenty-seventh SIGCSE technical symposium on Computer science education SIGCSE '96*, 237 – 241.

Deci, E. & Ryan, R. (1985). *Intrinsic motivation and self-determination in human behavior,* New York: Plenum Press.

www.manaraa.com

DeClue, T. (2008). Computer science in Kindergarten? Of course! The ABCs of the K-12 CSTA model curriculum in computer science. *Journal of Computing Sciences in Colleges, 23*(4), 257 – 262.

De Palma, P. (2001). Why women avoid computer science. *Communications of the ACM, 44*(6), 27 – 29.

Dobson, J. (2008). The use of formative online quizzes to enhance class preparation and scores on summative exams. *Advances in Physiological Education, 32*, 297-302.

Doerschuk, P., Liu, J., & Mann, J. (2007). Pilot summer camps in computing for middle school girls: From organization through assessment. *Proceedings of the 12th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education,* 4 – 8.

Downey, J. (2006). Measuring general computing self-efficacy: The surprising comparison of three instruments in predicting performance, attitudes, and usage. *Proceedings of the 39th Hawaii International Conference on System Sciences.*

Duvall, S. (2008). Computer science fairy tales. *Journal of Computing Sciences in Colleges, 24*(2), 98 – 104.

Eastman, E. (2003). Fact-based problem identification precedes problem solving. *Journal of Computing Sciences in Colleges, 19*(2), 18 – 29.

Edmondson, C. (2008). Real women don't write programs. *ACM SIGCSE Bulletin, 40*(2), 112 – 114.

Ericson, B., Guzdial, M., & Biggers, M. (2007). Improving secondary CS education: Progress and problems. *Proceedings of the ACM SIGCSE technical symposium on Computer Science Education '07,* 298 – 301.

Faulkner, K. & Palmer, E. (2009). Developing authentic problem solving skills in introductory computer science courses. *Proceedings of the ACM SIGCSE Technical Symposium on Computer Science Education '09,* 4 – 8.

Gay, L.R., Mills, G.E., & Airasian, P. (2006). *Educational research: Comptencies for analysis and applications* (8th ed.). Upper Saddle River, NJ: Pearson Education Inc.

Gay, L.R., Mills, G.E., & Airasian, P. (2009). *Educational research: Competencies for analysis and applications* (9th ed). Upper Saddle River, NJ: Pearson Education Inc.

www.manaraa.com

Gibson, J. & O'Kelly, J. (2005). Software engineering as a model of understanding for learning and problem solving. *Proceedings of the 2005 international workshop on Computing education research ICER '05*, 87 – 97.

Goold, A. & Rimmer, R. (2000). Indicators of performance in first-year computing. *Proceedings of the IEEE Computer Science Conference 2000. 23rd Australasian,* 74 – 80.

Guiller, J. & Durndell, A. (2007). Students' linguistic behavior in online discussion groups: Does gender matter? *Computers in Human Behavior, 23*(5), 2240 - 2255.

Hanks, B. & Brandt, M. (2009). Successful and unsuccessful problem solving approaches of novice programmers. *Proceedings of the ACM SIGCSE Technical Symposium on Computer Science Education '09,* 24 – 28.

Hardy, N. (2008). Women in computer science: Harnessing the power of web 2.0 to draw women to computer science fields. *Proceedings of the 9th ACM SIGITE conference on Information Technology Education,* 59 – 60.

Hart, M., Early, J., & Brylow, D. (2008). A novel approach to K-12 CS education: Linking mathematics and computer science. *Proceedings of the ACM SIGCSE technical symposium on Computer Science Education '08,* 286 – 290.

Hazzan, O., Gal-Ezer, J., & Blum, L. (2008). A model for high school computer science education: The four key elements that make it! *Proceedings of the ACM SIGCSE technical symposium on Computer Science Education '08,* 281 – 285.

Hu, H. (2008). A summer programming workshop for middle school girls. *Journal of Computing Sciences in Colleges, 23*(6), 194 – 202.

Jin, W. (2008). Pre-programming analysis tutors help students learn basic programming concepts. *Proceedings of the ACM SIGCSE technical symposium on Computer Science Education '08,* 276 – 280.

Joiner, R., Messer, D., Littleton, K., & Light, P. (1996). Gender, computer experience and computer-based problem solving. *Computers Education, 26*, 179 – 187.

Joseph, D. (2008). Increasing the number of entrants into the IT profession: The role of experiential training. *Proceedings of the ACM SIGCSE technical symposium on Computer Science Education '08,* 2 – 4.

Joshi, K.D. & Schmidt, N.L. (2006). Is the information systems profession gendered? Characterization of IS professional and IS careers. *The DATA BASE for Advances in Information Systems, 37*(4), 26 – 41.

المنارة للاستشارات
www.manaraa.com

Joyce, D. (1998). The computer as a problem solving tool: A unifying view for a non-majors course. *Proceedings of the twenty-ninth SIGCSE technical symposium on Computer science education SIGCSE '98*, 63 – 67.

Kibble, J.D. (2011). Voluntary participation in online formative quizzes is a sensitive predictor of student success. *Advances in Physiology Education, 35*(1), 95-96.

Kiesler, S., Sproull, L, & Eccles, J. (2002). Pool halls, chips, and war games: Women in the culture of computing. *ACM SIGCSE Bulletin, 34*(2), 159 – 164.

Klappholz, D. (2009). Organizing and delivering "real projects for real clients" courses. *Journal of Computing Sciences in Colleges, 24*(3), 106 – 108.

Klawe, M. & Leveson, N. (1995). Women in computing: Where are we now? *Communications of the ACM, 38*(1), 29 – 35.

Klawe, M. & Shneiderman, B. (2005). Crisis and opportunity in computer science. *Communications of the ACM, 48*(11), 27 – 28.

Kumar, A.N. (2008). The effect of using problem-solving software tutors on the self-confidence of female students. *Proceedings of the ACM SIGCSE technical symposium on Computer Science Education '08,* 523 – 527.

Lai, Y., & Wong, T. (2007). Infusing problem solving skills into computer lessons. *ACM SIGCSE Bulletin, 39*(4), 84 – 86.

Lau, W., Ngai, G., Chan, S., & Cheung, J. (2009). Learning programming through fashion and design: A pilot summer course in wearable computing for middle school students. *Proceedings of the ACM SIGCSE Technical Symposium on Computer Science Education '09,* 504 – 508.

Lee, J. S. (2008). Technology education for women. *Proceedings of the 2008 Conference on Computer Human Interaction,* 3447 – 3452.

Lei, S., Bartlett, K., Gorney, S., & Herschbach, T. (2010). Resistance to reading compliance among college students: Instructors' perspectives. *College Student Journal, 42*(2), 219-229.

Lepper, M., Greene, D., & Nisbett, R. (1973). Undermining children's intrinsic interest with extrinsic rewards: A test of the overjustification hypothesis. *Journal of Personality and Social Psychology, 28*(1), 129-137.

المنارة للاستشارات

www.manaraa.com

Lemire, D. (2002). Math problem solving and mental discipline – The myth of transferability. *Journal of College Reading and Learning, 32*(2), 229 – 238.

Lewis, T., Smith, W., Belanger, F., & Harrington, K. (2008). Determining students' intent to stay in IT programs: An empirical model. *Proceedings of the ACM SIGMIS-CPR '08 Conference,* 5 – 11.

Liu, M., Hsieh, P., Cho, Y., & Schallert, D. (2006). Middle school students' self-efficacy, attitudes, and achievement in a computer-enhanced problem-based learning environment. *Journal of Interactive Learning Research, 17*(3), 225 – 242.

Lopez, A., Jr., Schulte, L., & Giguette, M. (2005). Climbing onto the shoulders of giants. *Proceedings of the ACM SIGCSE Technical Symposium on Computer Science Education '05*, 401 – 405.

Lopez, A., Zhang, K., & Lopez, F. (2008). Cultural representations of gender among U.S. computer science undergraduates: Statistical and data mining results. *Proceedings of the ACM SIGCSE Technical Symposium on Computer Science Education '08,* 407 – 411.

McInerney, C., DiDonato, N., Giagnacova, R., & O'Donnell, A. (2006). Students' choice of information technology majors and careers: A qualitative study. *Information Technology, Learning, and Performance Journal, 24*(2), 35 – 53.

Madigan, E.M., Goodfellow, M., & Stone, J.A. (2007). Gender, perceptions, and reality: Technological literacy among first-year students. *Proceedings of the ACM SIGCSE Technical Symposium on Computer Science Education '07,* 410 – 414.

Maloney, J., Peppler, K., Kafai, Y., Resnick, M., & Rusk, N. (2008). Programming by choice: Urban youth learning programming with Scratch. *Proceedings of the ACM SIGCSE Technical Symposium on Computer Science Education '08,* 367 – 371.

Mayall, H. J. (2008). Difference in gender based technology self-efficacy across academic levels. *International Journal of Instructional Media, 35*(2), 145 – 155.

Moorman, P. & Johnson, E. (2003). Still a stranger here: Attitudes among secondary school students towards computer science. *Proceedings of the ACM ITiCSE '03 Symposium*, 193 – 197.

Moskal, B., Lurie, D., & Cooper, S. (2004). Evaluating the effectiveness of a new instructional approach. *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education,* 75 – 79.

المنارة للاستشارات
www.manaraa.com

Mullins, P., Whitfield, D., & Conlon, M. (2008). Using Alice 2.0 as a first language. *Journal of Computing Sciences in Colleges, 24*(3), 136 – 143.

Murphy, L. & Thomas, L. (2008). Dangers of a fixed mindset: Implications of self-theories research for computer science education. *Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education,* 271 – 275.

Myers, M. & Beise, C. (2001). Nerd work: Attractors and barriers perceived by students entering the IT field. *Proceedings of the ACM SIGCPR 200*, 201 – 204.

Nauta, M. M, & Epperson, D. L. (2003). A longitudinal examination of the social-cognitive model applied to high school girls' choices of nontraditional college majors and aspirations. *Journal of Counseling Psychology, 50*(4), 448 – 457.

Norris, C., Barry, F., Fenwick, J., Reid, K., & Rountree, J. (2008). ClockIt: Collecting quantitative data on how beginning software developers really work. *Proceedings of the 2008 ITiCSE Conference,* 37 – 41.

Olivieri, L. (2005). High school environments and girls' interest in computer science. *SIGCSE Bulletin, 37*(2), 85 – 88.

Olsen, A. (2005). Using pseudocode to teach problem solving. *Journal of Computing Sciences in Colleges, 21*(2), 231 – 236.

Owens, J. & Matthews, J. (2008). CyberCivics: A novel approach to reaching K-12 students with the social relevance of computer science. *Proceedings of the ACM SIGCSE technical symposium on Computer Science Education '08,* 372 – 376.

Palumbo, D.B. (1990). Programming language / problem-solving research: A review of relevant issues. *Review of Educational Research, 60*(1), 65 – 89.

Papastergiou, M. (2008). Are computer science and information technology still masculine fields? High school students' perceptions and career choices. *Computers & Education, 51*(2), 594 - 608.

Paxton, J. & Mumey, B. (2001). Teaching advanced problem solving: Implications for the CS curriculum. *Journal of Computing Sciences in Colleges, 16*(2), 51 – 56.

Pintrich, P. & Schunk, D., (1996). *Motivation in education: Theory, research, & applications.* Englewood Cliffs, NJ: Prentice Hall.

المنارة للاستشارات

www.manaraa.com

Pollock, L., McCoy, K., Carberry, S., Hundigopal, N., & You, X. (2004). Increasing high school girls' self confidence and awareness of CS through a positive summer experience. *Proceedings of the ACM SIGCSE technical Symposium on Computer Science Education '04*, 185 – 189.

Powell, R. M. (2008). Improving the persistence of first-year undergraduate women in computer science. *Proceedings of the ACM SIGCSE Technical Symposium on Computer Science Education '08*, 518 – 522.

Pulimood, S.M. & Wolz, U. (2008). Problem solving in community: A necessary shift in CS pedagogy. *Proceedings of the ACM SIGCSE Technical Symposium on Computer Science Education '08,* 210 – 214.

Quade, A. (2003). Development and validation of a computer science self-efficacy scale for CS0 courses and the group analysis of CS0 student self-efficacy. *Proceedings of the International Conference on Information Technology: Computers and Communications (ITCC'03),* 60 – 64.

Rafieymehr, A. (2008). Kids in computing (K.I.C.): Is there a solution to solve the computer science enrollment problem? *ACM SIGCSE Bulletin, 40*(2), 107 – 111.

Ragonis, N. & Hazzan, O. (2008). Tutoring model for promoting teaching skills of computer science prospective teachers. *Proceedings of the 2008 ITiCSE Conference,* 276 – 280.

Ramalingam, V. & Wiedenbeck, S. (1998). Development and validation of scores on a computer programming self-efficacy scale and group analysis of novice programmer self-efficacy. *Journal of Educational Computing Research, 19*(4), 367 – 381.

Rao, M.R.K. (2006). Storytelling and puzzles in a software engineering course. *Proceedings of the ACM SIGCSE Technical Symposium on Computer Science Education '06,* 418 – 422.

Reed, D. (2002). The use of ill-defined problems for developing problem-solving and empirical skills in CS1. *Journal of Computing Sciences in Colleges, 18*(1), 121 – 133.

Rieksts, I. & Blank, G. (2008). Inspiring future IT professionals with Mars rovers. *Journal of Computing Sciences in Colleges, 23*(5), 44 – 51.

Rocca, K.A. (2004). College student attendance: Impact of instructor immediacy and verbal aggression. *Communication Education, 53,* 185 – 195.

المنارة للاستشارات

www.manaraa.com

Rosser, S. V. (2005). Women and ICT: Global issues and actions. *Proceedings of the international symposium on Women and ICT.*

Sackrowitz, M. G. & Parelius, A. P. (1996). An unlevel playing field: Women in the introductory computer science courses. *Proceedings of the ACM SIGCSE technical Symposium on Computer Science Education '96*, 37 – 41.

Sands, M., Moukhine, N., & Blank, G. (2008). Widening the pipeline of K-12 students with Flash. *Journal of Computing Sciences in Colleges, 23*(5), 52 – 57.

Scragg, G. & Smith, J. (1998). A study of barriers to women in undergraduate computer science. *Proceedings of the ACM SIGCSE technical Symposium on Computer Science Education '98*, 82 – 86.

Sivilotti, P.A.G. & Laugel, S.A. (2008). Scratching the surface of advanced topics in software engineering: A workshop module for middle school students. *Proceedings of the ACM SIGCSE technical symposium on Computer Science Education '08,* 291 – 295.

Sloan, R.H. & Troy, P. (2008). CS 0.5: A better approach to introductory computer science. *Proceedings of the ACM SIGCSE technical Symposium on Computer Science Education '08,* 271 – 275.

Sullivan, F. R. & Lin, X. (2006). The ideal science student and problem solving. *Proceedings of the 7th International Conference on Learning Sciences,* 737 – 743.

Takruri-Rizk, H., Jensen, K., & Booth, K. (2008). Gendered learning experience of engineering and technology students. *SIGCAS Computers and Society, 38*(1), 40 – 52.

Tang, M., Pan, W., & Newmeyer, M. D. (2008). Factors influencing high school students' career aspirations. *Professional School Counseling, 11*(5), 285 – 295.

Teague, J. (2002). Women in computing: What brings them to it, what keeps them in it? *ACM SIGCSE Bulletin, 34*(2), 147 – 158.

Townsend, G., Barker, L., Menzel, S., & Cohoon, J. (2008). Grace Hopper visits the neighborhood. *Proceedings of the ACM SIGCSE Technical Symposium on Computer Science Education '08,* 513 – 517.

Treu, K. & Skinner, A. (2002). Ten suggestions for a gender-equitable CS classroom. *ACM SIGCSE Bulletin, 34*(2), 165 – 167.

www.manaraa.com

Tu, J. & Johnson, J. (1990). Can computer programming improve problem-solving ability? *ACM SIGCSE Bulletin, 22*(2), 30 – 37.

U.S. Department of Education, National Center for Education Statistics. (2007). Digest of Education Statistics 2007. Retrieved September 12, 2008, from http://nces.ed.gov/programs/digest/d07/tables/dt07_265.asp.

Van Sickle, E. (2008). Refilling the IT pipeline and using storage technologies as a specialization. *Proceedings of the ACM SIGMIS-CPR 2008 Conference,* 112 – 118.

Whimbley, A. & Lochhead, J. (1999). *Problem solving & comprehension. 6th Edition.* Mahwah, N.J.: Lawrence Erlbaum Associates.

Wilson, B.C. (2006). Gender difference in types of assignments preferred: Implications for computer science instruction. *Journal of Educational Computing Research, 34*(3), 245 – 255.

Wilson, B. (2008). Improving comfort level of females in the first computer programming course: Suggestions for CS faculty. *Journal of Computing Sciences in Colleges, 23*(4), 28 – 34.

المنارة للاستشارات

www.manaraa.com