DOCUMENT RESUME

ED 303 153 IR 013 623

AUTHOR Barger, Robert N.
TITLE On-Line Evaluation and Remediation of Programming Skills.
PUB DATE Apr 88
NOTE 8p.; Paper presented at the Annual Meeting of the International Association for Computing in Education (New Orleans, LA, April 5-9, 1988).
PUB TYPE Guides - Classroom Use - Guides (For Teachers) (052) -- Computer Programs (101) -- Speeches/Conference Papers (150)

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS *Computer Assisted Testing; *Evaluation Criteria; *Programing; Programing Languages; Psychometrics; Remedial Instruction; *Student Evaluation; Test Construction

ABSTRACT

This procedure for testing the mastery of programming skills uses online testing and correction. The student is presented with a test problem and is asked to solve it, encode the solution, debug the code, and save it on a floppy disk. The instructor corrects the program at the appropriate points through the use of REM statements. The erroneous or inefficient student-written lines are neutralized but retained in the REM statements so that they can be compared by the student with the correct lines. A variant on this procedure would have the instructor point out the place where an error has occurred, but the student would make the corrections and resubmit the test. Two sample questions with corrections in the BASIC Programming Language are provided, although the procedure could be adapted to many computer languages. (EW)

ON-LINE EVALUATION AND REMEDIATION OF PROGRAMMING SKILLS

by

Robert N. Barger, Ph.D.
Professor of Education
Eastern Illinois University
Charleston, Illinois 61920

A paper presented at the 1988 Annual Meeting of
the International Association for Computing in Education
New Orleans, April 5-9, 1988

The procedure described below is adaptable for use in testing mastery of programming skills in many computer languages. For purposes of demonstration, the procedure will be outlined here as it might be applied to testing mastery of programming skills in the BASIC language.

In this procedure, instead of employing the traditional practice of testing the student in off-line paper and pencil mode, the student is presented with a test problem and is asked to solve it, encode the solution on-line, debug the code, and save it on the student's floppy disk. The instructor then collects the disk and runs and corrects the test program at his/her leisure. This is done by inserting corrections in the program at appropriate points through the use of REM statements. The corrections change erroneous or inefficient lines to executable or more efficient lines. The original errors or inefficiencies are neutralized and retained in REM statements on the line before the corrected version so that the student can compare his/her original statements with the instructor's revisions. Additional instructor comments, along with the student's grade on the test, can be put in REM statements at the beginning or end of the program. When the instructor is finished evaluating and annotating the program, it is re-saved on the floppy disk in it's revised form and returned to the student for review. If the instructor so desires, he/she can also save the student's revised marked-up program on an archive disk before returning the student disk, and/or print out a hard copy of the program.

A variant of the above-described procedure would have the instructor use REM statements to point out where errors have occurred in the test program. Instead of supplying the corrected version by rewriting the erroneous lines or supplying any missing lines, the instructor may have the student make the corrections and resubmit the test - perhaps giving some suggestions, if it seems indicated, on how this might be done.

This evaluation and remediation procedure is supported by two psychometric principles: 1) If the skill to be acquired in the course involves the ability to do programming, then it is that skill which should be tested, rather than testing how well the student can memorize lecture or textbook material. Thus, in this proposed testing procedure, the student is asked to actually write a program, rather than to write about how to program, and 2) real-world conditions are simulated, that is, the student has a time limit imposed (much as there would be a deadline in real life), but the use of resources such as notes, texts, and documentation is allowed (since these materials would be available for reference in real-world situations). Also, when possible, the programming problem assigned should have some evident real-world application (the sample questions attached are not good illustrations of this latter point).

Two sample questions (one a lab assignment and the other a periodic test), along with corrected responses to the questions as handed back to the students who authored them, are given on the attached pages for purposes of illustration.

LAB8 ASSIGNMENT

SED 1099 LAB8 DR. BARGER

WRITE A PROGRAM THAT ACCEPTS AS INPUT A USER'S AGE AND THEN USES A WHILE/WEND LOOP TO COMPUTE HOW MANY YEARS IT WILL BE BEFORE THAT PERSON IS ONE HUNDRED YEARS OLD. THE OUTPUT SHOULD INFORM THE USER HOW MANY YEARS IT WILL BE UNTIL HE/SHE IS ONE HUNDRED.

LAB8 PROGRAM LISTING

[INSTRUCTORS COMMENTS IN LINES 1 - 9]

```
1 ' I ADDED LINE 11 TO CLEAR THE SCREEN AND TURN THE KEY OFF.
2 ' IN LINE 20, YOUR INPUT LABELLING SHOULD MAKE IT CLEARER TO THE USER WHAT
    KIND OF INPUT YOU ARE EXPECTING, AND YOUR CHOICE OF A VARIABLE NAME SHOULD
    BE MORE MEANINGFUL. SEE CORRECTION IN LINE 21.
3 ' IN LINE 30, THERE IS NO NEED TO INITIALIZE A VARIABLE NAMED `COUNTER'
    SINCE YOU DON'T USE IT LATER IN THE PROGRAM.
4 ' IN LINE 40, THE `WHILE' STATEMENT SHOULD TEST IF THE INPUT VARIABLE IS LESS
    THAN 100. ALSO, YOU SHOULD KEEP VARIABLE NAMES CONSTANT BETWEEN LINES 20 AND
    40. SEE CORRECTION IN LINE 41.
5 ' IN LINE 50, THE VARIABLE NAME `NO.YEARS' SHOULD BE A BIT MORE MEANINGFUL.
    SEE CORRECTION IN LINE 51.
6 ' IN LINE 60, YOUR ACCUMULATOR SHOULD BE FORMULATED ACCORDING TO THE
    CORRECTED VERSION IN LINE 61.
7 ' THE FORMAT OF YOUR PRINT USING STATEMENT IN LINE 90 IS INCORRECT. SEE
    CORRECTION IN LINE 91.
8 ' IN LINE 100, IN CREATING A LOOP YOU MUST INITIALIZE THE COUNTER `YRS.TO.100'
    BEFORE YOU START THE LOOP. SEE CORRECTION IN LINES 101 AND 102.
9 ' LAB8 SCORE = 1 [OUT OF A POSSIBLE 2].
10 '[STUDENT'S NAME APPEARS ON THIS LINE], LAB8
11 CLS:KEY OFF
20          'INPUT "YEARS OLD";YO
21 INPUT "HOW OLD ARE YOU NOW";AGE
30          'COUNTER = 0
40          'WHILE NUMBER.OF.YEARS = 0
41 WHILE AGE < 100
50          'NO.YEARS = NO.YEARS + 1
51   YRS.TILL.100 = YRS.TILL.100 + 1
60          'TOTAL = 19 + 81
61   AGE = AGE + 1
80 WEND
90          'PRINT USING "AFTER ## YEARS ,BEFORE ###";NO.YEARS
91 PRINT "AFTER";YRS.TILL.100;"YEARS YOU WILL BE 100 YEARS OLD."
100         'GOTO 20
101 YRS.TILL.100 = 0
102 GOTO 20
110 END
```

TEST3 QUESTION

SED 1099 TEST3 DR. BARGER

START YOUR PROGRAM WITH A 'REM' STATEMENT CONTAINING YOUR NAME. THEN ADD A LINE WITH A STATEMENT TO CLEAR THE SCREEN AND TURN THE KEY OFF. THEN WRITE PROGRAM LINES WHICH USE A 'GOTO' LOOP AND 'READ' AND 'DATA' STATEMENTS TO READ A SERIES OF PRICES (GIVEN BELOW), AS EACH PRICE IS READ ADD IT TO AN ACCUMULATOR. ALSO, USE A COUNTER TO KEEP TRACK OF HOW MANY PRICES HAVE BEEN READ. THEN (WHILE STILL WITHIN THE LOOP) PRINT THE AVERAGE OF THE PRICES READ SO FAR (DO THIS BY DIVIDING THE ACCUMULATOR BY THE COUNTER). THE AVERAGE SHOULD BE PRINTED USING ['PRINT USING'] THE FOLLOWING FORMAT: $##.## USE THE FOLLOWING DATA FOR THE PRICES: 3.146, 2.709, 67.1, 0.9328, 77.034, 5.45754. DON'T FORGET THE 'END' STATEMENT. THE OUTPUT SHOULD LOOK LIKE THIS:

```
$ 3.15
$ 2.93
$24.32
$18.47
$30.18
$26.06
Out of Data in 30
```

TEST3 PROGRAM LISTING

[INSTRUCTORS COMMENTS IN LINES 1 - 7]

```
1 ' LINE 50 SHOULD BE PLACED JUST BEFORE 60 AND SHOULD SAY SOMETHING LIKE:
 REM READ STATEMENT FOR DATA.  LINES 56 AND 60 SHOULD BE REVERSED - YOU MUST
 FIRST READ THE PRICE AND THEN ADD IT TO THE ACCUMULATOR.
2 ' ALSO, YOU MUST ADD THE PRICE TO WHAT IS ALREADY IN THE ACCUMULATOR -
 `ACCUMULATOR = 0 + PR' DOESN'T DO THIS, IT HAS TO BE:  ACCUMULATOR =
 ACCUMULATOR + PR
3 ' LINE 90 IS NOT NEEDED AND SHOULD BE OMITTED. LINE 100 SHOULD PRINT `AVE.PR',
 NOT `PR'. THE `GOTO' STATEMENT ON LINE 105 SHOULD LOOP BACK TO LINE 50 - NOT
 LINE 40, OTHERWISE THE ACCUMULATOR WILL BE INITIALIZED AGAIN.
4 ' LINES WITHIN THE LOOP SHOULD BE INDENTED FOR EASE IN IDENTIFYING WHAT IS
 GOING ON INSIDE THE LOOP.
5 ' SEE CORRECTED VERSION OF PROGRAM IN LINES 500-
6 ' TEST 3 = 6 POINTS   [OUT OF A POSSIBLE 11]
7 ' PRESENT AVERAGE:   PRINT 1.8+1.6+0.7+1.25+1.6+2+2+1.15+   7+10+6
    = 35.1 / 11  = .7163265 = 72% = a high "D"
10 ' [STUDENT'S NAME APPEARS HERE], TEST3
20 CLS: KEY OFF
30 COUNTER = 0
40 ACCUMULATOR = 0
50 REM *** INPUT DATA FOR READ STATEMENT ***
55 COUNTER = COUNTER + 1
56 ACCUMULATOR = 0 + PR
60 READ PR
70 AVE.PR = ACCUMULATOR / COUNTER
90 PRINT "PRICE"; PR; ACCUMULATOR
100 PRINT USING "$##.##";PR
105 GOTO 40
110 REM *** DATA USED BY READ STATEMENT ***
120 DATA 3.146,
130 DATA 2.709
140 DATA 67.1
150 DATA 0.9328
160 DATA 77.034
170 DATA 5.45754
180 END
500 ' $%$%$%  CORRECTED VERSION  $%$%$%
510 CLS:KEY OFF
520 TOTAL.PRICE = 0
530 COUNTER= 0
540 READ PRICE
550  COUNTER = COUNTER + 1
560  TOTAL.PRICE = TOTAL.PRICE + PRICE
570  AVERAGE = TOTAL.PRICE / COUNTER
580  PRINT USING "$##.##";AVERAGE
590 GOTO 540
600 DATA 3.146,2.709,67.1,0.9328,77.034,5.45754
610 END
```