The University of Wisconsin
Computer Sciences Department
1210 West Dayton Street
Madison, Wisconsin 53706

# A GENERATIVE CAI PROGRAM THAT TEACHES ALGEBRA

by

Carol Peplinski

Technical Report #90

April 1970

# A GENERATIVE CAI PROGRAM THAT TEACHES ALGEBRA[1]

by

Carol Peplinski[2]

## Introduction

Computer-assisted instruction is a field with many and varied possibilities, several of which are only gradually becoming apparent. One of these is a system which generates questions to be asked of a student, then compares the student's answer with its own and tells him whether he is right or wrong. This sort of system allows the student to do as many problems as necessary to master the subject matter in question and further gives him immediate knowledge of results. Particular problems, answers, and branches do not have to be pre-programmed. Instead, the computer computes the problem, and computes its answer. Thus a far greater number of problems, and of branches, are potentially available.

A generation program can be built most easily when the subject is rather basic, with obvious logical connections. Uhr[1,2] describes a program which generates elementary arithmetic problems as well as problems involving simple transformations, e.g., from one language to another. Suppes3 gives a good example of the more traditional approach, describing a system giving elementary school children practice in basic relationships between numbers.

This paper describes a drill and practice system which generates problems at a somewhat different level, first year algebra. It does not attempt to teach the student from the beginning, but rather assumes that he has heard an explanation of

---

[1]This research was supported in part by NSF grants GP-7069 and GJ-583.

[2]Now at Bell Telephone Laboratories, Chicago.

how to do the problems in question. It offers its own explanation only when the student has given a wrong answer. If he never answers incorrectly, the student will be given no explanations. The program aims to give its users practice in working algebra problems with explanation when necessary, but does not present an entire algebra course.

## Program Description

### Overview

The system begins by asking the student for his name, then follows with questions regarding the type of problem the student desires to work. He replies by typing either one of several designations of type or the word OPTION, which results in a description of kinds of problems generated and their respective calls. When the student has made his choice, the computer generates an algebra problem to which it expects an answer. As soon as it has received one, or in the case of quadratic equations, two, it compares the student's answer to that generated along with the equation. If they agree, another problem is generated; if they disagree, the program explains a method of solution and generates another problem. After a predetermined number of problems (five at present) have been worked, the program compares the number right to the total number generated. If eighty per cent or more of the problems have been solved correctly, the program generates more difficult problems, degree of difficulty being determined by the size of numbers in the equations. This continues until eighty per cent of the most difficult problems of the specified class have been answered correctly. The student is then asked whether he would like to do more problems; if so, he can type the appropriate call and continue.

The teacher supervising the use of the program has the option of writing his own explanation for any given type of problem and inserting it into the running system at any time. The system then presents this instead of the built-in explanation.

An internal record is automatically kept of each student's success with each kind of problem. The number done correctly is stored under each student's name, so that the information is available for the teacher's scrutiny at any time upon reception by the computer of the proper calling word, RESULTS.

Detailed Description

The system as it now stands generates either linear or quadratic equations. The linear equations can be divided roughly into two groups: those with x-terms on one side of the equals sign and those with x-terms on both sides. When there are x-terms on only one side, there is also an option of having the answers limited to integers. In all types of linear equations, the student determines how many x-terms will appear on each side of the equals sign. If the number chosen is greater than 1, the student will gain experience in combining terms. If the number is 1, the problem will be of form $ax + b = c$ or $ax + b = cx + d$, depending on the option chosen. In any case the program begins by generating problems with only one x-term on each side and gradually adds more as the student solves the easier ones, until finally the specified maximum has been reached.

All of the quadratic equations now being generated by the system are solved by factoring into a form specified by the call. Options include equations to be solved by taking the square root of both sides or by completing the square on the

left and then taking the square root, equations which are the difference of two squared terms, and basic factoring problems of the form $ax^2 + bx + c = 0$. In each case, the coefficient of the $x^2$ term may be specified to be 1, or the equation may require combination of terms to be reduced to one of the above forms.

As soon as the program has identified which type of problem it has been asked to generate, it branches to the appropriate section of code. If it is to generate linear equations, it sets the maximum size of coefficients, then calls the function GENERATE, which returns a combination of x-terms and constants obtained from the random number generator together with the sum of the x-terms and constants. Each output of GENERATE may be used as one side of an equation, so the function is called twice if the student has specified x-terms on both sides of the equation, once if on only one side. If there are x-terms on only one side, a random number is generated for the other side. The program keeps track of how many x-terms there are to be on each side, beginning with one and gradually adding them until the specified number has been reached.

The returned totals of x-terms and constants are then used to compute the answer to the problem. This is stored in four separate variables, one each for the sign, the integer answer, the numerator, and the denominator of a fraction. After the answer has been computed, the problem appears before the student on an output device. When the student's answer has been read, it too is broken into four variables, each of which is then compared to those calculated by the computer. If all agree, the student is told he has obtained the correct answer (the actual wording of this varies from problem to problem in a random manner) and the program

branches to generate the next problem. If either the sign or the integer portion of the answers disagree, the program branches to an explanation immediately with the function comparing the answers returning the value of the correct answer in a form suitable for use in the explanation. Disagreement between the fractional portions of the answers requires further tests since the computer often arrives at answers which are not in lowest terms.

The explanation given is basically the same no matter what kind of linear equation has been generated. Tests are made, however, so that sections irrelevant to the specific problem are omitted. For example, if there is only one x-term on each side, the instruction to combine terms on each side of the equation is bypassed.

The section of code which generates quadratic equations begins by setting a series of indicators which tell the system which combination of options has been chosen. For example, if the call is DIFFSQSIM, problems which are the difference of two squares with the coefficient of the $x^2$ term not necessarily 1 and require simplification before they can be solved, an indicator is set to show that the problem is to require simplification. Another, the "square" indicator, says that both answers are to have the same absolute value. Others indicate that the equation is to factor into a sum and a difference and that the coefficient of $x^2$ need not be 1.

After all indicators have been set, the program begins generating the equation itself. It chooses numbers a,b,c, and d at random within the constraints of the indicators so that the equation is of form $(ax + b)(cx + d) = 0$, then multiplies out the left side. In the example DIFFSQSIM, it would choose a at random (absolute value less than the upper limit placed on coefficients). Then, since the "square"

indicator shows that the answers are to have the same absolute value, c is set to equal a. Next b is generated, and since another indicator shows that the answers are to have opposite signs, d is set to equal -b, and the numbers are multiplied to give the left side of a quadratic equation, $a^2x^2 - b^2$. The variable holding the right side of the equation is set to 0 and both answers are calculated by a function called for the purpose. They are returned in the same four variable form as the answers to the linear equations.

Then, since an indicator shows that the equations are to require simplification before factoring, the program branches to a section which adds random amounts to each term on both sides of the equation, leaving the answer unchanged.

Any one of the calls would be treated in a manner similar to DIFFSQSIM. SIMPLIFY does not require that both answers have the same absolute value, so b and d would be chosen independently of one another. Also, a would be chosen at random with c equaling 1. Later c would also be chosen at random. SQSIM would be identical to DIFFSQSIM except that both answers have the same sign as well as the same absolute value COMPLSQSIM would involve the subtraction of the square of a number randomly chosen from the left side of the equation with appropriate adjustments to the stored answers.

When the equation has been placed in the form necessary for output, it is compared to the last equation presented to the student and replaced by another equation generated in its place if it is the same. This process is necessary mainly in the case of simple equations where both roots have the same absolute value since they require only one random number and the random number generator may easily

produce the same number twice in succession. After the equation has passed this test it is presented to the student.

The process of checking to see whether the student's answers are correct is identical with the one used for linear equations. If a wrong answer is given, however, there are several possible explanations, depending on which kind of problem is being worked. After like terms have been combined, the program branches to explain what kind of problem is in question and how it can be recognized. The equation is factored, answers are given, and another problem of the same type is generated.

An explanation for each sort of problem generated is written into the system and will be used in response to an incorrect answer in the absence of other specification. If the teacher supervising the drill and practice is unhappy with a particular explanation, he may input a new one for any given type or types of problem. The revised explanation consists of several lines, each of which may be labeled or unlabeled, and may consist of a combination of literals, which will be printed exactly as they are read, and certain specified variables depending on the type of problem with which the explanation is to be used. These variables consist mainly of the various values of terms in the generated equations and the answers to them. It is also possible to erase an explanation, reverting to that written into the system, by reading in a null explanation.

To enter a new explanation, e.g., a new explanation for FACTOR, equations of the form $ax^2 + bx + c = 0$, one first types "*FACTOR" when the machine asks for the name of the type of equation to be generated. The "*" tells the system that

what follows is to be a changed explanation rather than a call for it to generate problems. "*FACTOR" must be followed by at least two spaces. The asterisk must be found in column one and no intervening spaces should occur. The desired new explanation then follows.

If a line of the new explanation is to be labeled, "++label++", where label is some name, precedes the explanation proper. The label must contain at least one alphabetic character. Whether a label is used or not, the main body of the line of explanation follows, the end being marked by "//". Each line consists of a combination of literals and variables with the literals enclosed by quotes, e.g., 'THE ANSWER IS' SIDE1 (SIDE1 is a variable name.) Anything inside the quote marks is printed later exactly as read. Literals and variables may be mixed in any order within a line. Literals may be broken across cards but not across lines. Each line must be a complete unit. Variables must be followed by a space to mark their ending, but there is no real need for them to be preceded by one.

The variables allowed depend on the type of equation the input is to explain. In any sort of quadratic equation, variables to be used are

| | |
|---|---|
| A<br>B<br>C<br>D | where the equation is<br>$(Ax + B)(Cx + D)$ |
| FNT | equal to $A \cdot C$, the coefficient of $x^2$ |
| MID | equal to $B \cdot C$ $A \cdot D$, the middle term |
| ENDI | equal to $B \cdot D$, the last term |
| NI1 | value POSI for MID $\geq 0$<br>NEGA for MID $< 0$ |
| SIDE1<br>SIDE2 | the two solutions to the equation. |

In a linear equation allowed variables are

| | |
|---|---|
| XLTOT | total of x-terms on the left side |
| TOT | total of constants on the left |
| XRTOT | total of x-terms on the right side |
| RTOT | total of constants on the right |
| XTOT | total of all x-terms |
| CONTOT | total of all constant terms |
| SIDE1 | the answer |
| OUT1<br>OUT2 | left and right sides of the equation as originally generated. |

In both sorts of equation, the variable QUOTE is also allowed, providing a means of inserting a quote mark in a literal. The functions SIGN and MAKNUL may be called, with SIGN putting a + or - sign as appropriate in front of a constant or variable and MAKNUL returning a null value if its argument is 1, the value of its argument otherwise. (MANKUL is used to suppress the 1 in "1x").

A line to be printed, if read in on cards, may extend over as many cards as necessary. Also, as many lines as desired may be punched on one card. The end of the entire explanation is specified by "***" starting in the first column of the last card.

To eliminate an explanation which has been read in and return to that supplied by the system "***" should occur immediately following the two spaces required after the name of the equation type.

## Results

The system as so far described seems to handle the problems inherent in generating correct answers for its problems and then distinguishing right and wrong answers given it quite well. In simple problems where the answer is an integer, recognizing the correct solution presents no great difficulty. However when the answer is a fraction the situation may become more complicated. If the answer generated by the machine is not in lowest terms, several tests beyond simple comparison are necessary. The two answers required in a quadratic equation also complicate the situation since the program must be prepared to recognize them in either order and must not be confused if one answer is correct and the other is not. Figure 1 gives a partial listing of kinds of problems available as they are described to the student. Figure 2 presents interactions with simulated students, Figure 3 illustrates the use of the explanation revision option, and Figure 4 gives an example of the form in which results are presented to the teacher.

## Possible Improvements and Extensions

Several improvements to the system as now written are apparent. Something should be done to make sure that no equation reaches the student with all of its coefficients divisible by the same number. For example, the equation generated by FACTOR, $5x^2-5x-30=0$, should be reduced to $x^2-x-6=0$. It may be desirable to have a few equations of this sort left unchanged, but in that case the explanation should begin by instructing the student to divide both sides by 5. At present, all explanations factor the equations into the form in which they were generated. The example would thus be factored to $(5x+10)(x-3)$ rather than $(x+2)(x-3)$.

The need for another similar improvement becomes apparent in many cases when the correct answers are fractions. While the program recognizes fractions in lowest terms as correct, it does not reduce its own answers. Some routine should be added to make sure that all fractions handled have been reduced, thus forcing the student to reduce his answers.

Shortcomings also exist in the explanation revision routine. At the present time, the system does not allow branching within the explanations read in. Thus no use is made of the label. The explanations are stored line by line in a manner which should make branching easy to implement, but the tests necessary to recognize branching statements have not been programmed. This will have to include some sort of conditional branching, at least testing of equality, less than, and greater than.

It is possible to write explanations for the equations generated without branching without much difficulty. It is, however, more efficient to use part of the same explanation for several different kinds of equations. For example, once a quadratic equation has been factored, solving for the answers is the same no matter what the equation originally looked like, and has much in common with solving linear equations once their terms have been collected.

Another useful improvement would be a check for syntax errors in the users' explanations. At the moment, if a user were to break a literal across a line the program would go into an infinite loop. Most errors in demarkation of labels or lines will eventually lead to some sort of error message, but mistakes within a line may not stop the program at all. The addition of branching would open possibilities for many more kinds of errors and probably make some sort of checking routine a necessity.

The system at present assumes that a student will be able to work up to a level of doing eighty per cent of his problems correctly in one sitting. It might be a good idea to add a provision for the student to stop when he is tired and then restart at a later time, the level of difficulty, type, and number of problems yet to be done being stored by the computer. Something similar to the system described by Suppes, in which a student who is successful in less than sixty per cent of the problems he attempts moves down a level, might also be incorporated. This would call for an ordering of levels of difficulty among the different types of quadratic equations beyond what is now operational.

The descriptions of problems stored for the teacher's reference need improvement, since all levels of difficulty are stored under the same title. This is especially true of the linear equations.

## Discussion

The program described in this paper has its main similarities to those described by Uhr in the basic idea of generating questions and their answers rather than presenting pre-stored problems. Both Suppes and Uhr offer the student a second chance when he has once given a wrong answer. The described system does not, on the theory that a failure with an algebra problem is likely to be due either to an arithmetic error in combining terms or a lack of understanding of the method to be used. In the first case the error is not fundamental, should not often recur, and will probably take the student longer to find by checking back in his calculations than it merits. In the latter case any second answer will be nothing but guessing. The present system also makes no effort to store problems the student has been unable to work, as does Uhr's.

## Summary

The system described above presents algebra problems of various types, individual problems within each type becoming more difficult as more and more are done successfully. It reads the student's answers and tells him if they are correct, then generates another problem or explains the problem answered incorrectly, whichever is appropriate. It keeps a record of each student's successes and failures available for the teacher's reference upon demand, and allows the student to progress to the next level of difficulty when he has done eighty per cent of the problems presented on his level correctly.

The supervising teacher also has the option of revising or replacing explanations.

Suggested extensions include a wide range of problem types and an increased flexibility in the changes the supervising teacher is allowed to make in explanations. Increased possibilities of switching students from one level to another are also suggested.

## References

1. Uhr, L., The Automatic Generation of Teaching Machine Programs Center for Teaching and Learning Report, Ann Arbor, 1965.

2. Uhr, L., Teaching machine programs that generate problems as a function of interaction with students. Proc. 1969 Annual Meeting of the ACM, San Francisco, 1969, pp. 125-133.

3. Suppes, P., "The Uses of Computers in Education," Scientific American, 215 (September 1966) pp. 207-220.

Figure 1. Part of the explanation of problem types.

```
HELLO. LET'S DO SOME ALGEBRA PROBLEMS. FIRST TYPE YOUR NAME ON THE TYPEWRITER.
STUDENT
NOW TYPE THE NAME OF THE KIND OF PROBLEM YOU WISH TO DO. IF YOU WOULD LIKE AN EXPLANATION OF THE
POSSIBLE TYPES. TYPE
THE WORD OPTION.
OPTION

TYPE OF PROBLEM                                                    PHRASE TYPE

QUADRATIC EQUATIONS OF FORM
      A(1)X**2 + A(2)X + A(3) = 0 WHICH FACTOR TO
            (X-B)(X-D)                                             FACTOR1
            (AX-B)(CX-D)                                           FACTOR
            (X-B)(X-B)                                             SQ1
            (AX-B)(AX-B)                                           SQ
            (X-B)(X+B)                                             DIFFSQ1
            (AX-B)(AX+B)                                           DIFFSQ

      A(1)X**2 + A(2)X + A(3) = B(1)X**2 + B(2)X + B(3) WHICH FACTOR TO
            (X-B)(X-D)                                             SIMPLIFY1
            (AX-B)(CX-D)                                           SIMPLIFY
            (X-B)(X-B)                                             SQSIM1
            (AX-B)(AX-B)                                           SQSIM
            (X-B)(X+B)                                             DIFFSQSIM1
            (AX-B)(AX+B)                                           DIFFSQSIM

QUADRATIC EQUATIONS TO BE SOLVED BY COMPLETING THE SQUARE OF FORM

      A(1)X**2 + A(2) + A(3) = 0 WHICH FACTOR TO
            (X-B)(X-B) = E**2                                      COMPLSQ1
            (AX-B)(AX-B) = E**2                                    COMPLSQ
      A(1)X**2 + A(2)X + A(3) = B(1)X**2 + B(2)X + B(3) WHICH FACTOR TO
            (X-B)(X-B) = E**2                                      COMPLSQSIM1
            (AX-B)(AX-B) = E**2                                    COMPLSQSIM

NOW TYPE THE CALL FOR THE KIND OF PROBLEM YOU WOULD LIKE TO SOLVE.
```

• Figure 2. Problems and explanations generated by the program.
Linear equations.

```
-4X -1 +5X +4  = -3
YOU HAVE MADE AN ERROR. THE PROBLEM IS
-4X -1 +5X +4  = -3
FIRST COLLECT ALL TERMS CONTAINING ≠X≠ AND ALL CONSTANTS ON THE LEFT. THIS IS
1X +3 = -3
SUBTRACT 3 FROM BOTH SIDES, OBTAINING
1X = -6

+4X  = -4X -4
YOUR ANSWER IS -1/2
GENERATED ANSWER IS -0 4/8
CORRECT.

-3X +6  = +8X
YOUR ANSWER IS 11/12
GENERATED ANSWER IS 0 6/11

YOU HAVE MADE AN ERROR. THE PROBLEM IS
-3X +6  = +8X
SUBTRACT 8X FROM BOTH SIDES. THIS PRODUCES
-11X +6 = 0
SUBTRACT 6 FROM BOTH SIDES, OBTAINING
-11X = -6
NOW DIVIDE BOTH SIDES BY -11. AND X = 6/11
```

```
+8X +2 +5X -5 -5  = +2X +7 +3X +4 +6X -8
YOUR ANSWER IS 5 2/4
GENERATED ANSWER IS 5 1/2

YOU HAVE MADE AN ERROR. THE PROBLEM IS
+8X +2 +5X -5 -5  = +2X +7 +3X +4 +6X -8
FIRST COLLECT ALL TERMS CONTAINING ≠X≠ AND ALL CONSTANTS ON THE LEFT. THIS IS
 13X -8 = +2X +7 +3X +4 +6X -8
THEN DO THE SAME ON THE RIGHT. THIS RESULTS IN
13X -8 = 11X +3
SUBTRACT 11X FROM BOTH SIDES. THIS PRODUCES
2X -8 = 3
SUBTRACT -8 FROM BOTH SIDES, OBTAINING
 2X = 11
NOW DIVIDE BOTH SIDES BY 2. AND X = 5 1/2
```

Quadratic equations.

```
4X**2 +9X -9 = 0
YOUR ANSWERS ARE X = -3, X = 3/4
GENERATED ANSWERS ARE  X = 0 3/4, X = -3 0/1
CORRECT.
```

```
X**2 -11X +30 = 0
YOUR ANSWERS ARE X = 1, X = -1
FACTOR THE LEFT MEMBER INTO (X -5)(X -6)
SET EACH FACTOR EQUAL TO ZERO.
X -6 = 0
X -5 = 0
SOLVE THE RESULTING LINEAR EQUATION.
X = 5X = 6
```

```
4X**2 -3X -10 = 0
YOUR ANSWERS ARE X = 2, X = -1 1/4
GENERATED ANSWERS ARE  X = -1 1/4, X = 2 0/1
CORRECT.

.

4X**2 -25 = 0
YOUR ANSWERS ARE X = 2 1/3, X = -2 1/3
GENERATED ANSWERS ARE  X = -2 1/2, X = 2 1/2

YOU HAVE MADE AN ERROR. THE PROBLEM IS
 4X**2 -25 = 0
THE EXAMPLE IS A DIFFERENCE OF TWO SQUARES, AND
THUS FACTORS INTO THE SUM AND DIFFERENCE OF THE
SQUARE ROOTS, AS FOLLOWS
(2X +5)(2X -5) = 0
THEN 2X = -5 AND 2X = 5
DIVIDING TO SOLVE, X = -2 1/2 OR X = 2 1/2


 5X**2 +41X +78 = 2X**2 +7X -2
YOUR ANSWERS ARE X = -3 1/3, X = -3 1/3
GENERATED ANSWERS ARE  X = -3 1/3, X = -8 0/1

YOU HAVE MADE AN ERROR. THE PROBLEM IS
 5X**2 +41X +78 = 2X**2 +7X -2
FIRST COLLECT TERMS BY SUBTRACTING 2X**2, 7X, AND -2 FROM BOTH SIDES. THIS RESULTS IN
3X**2 +34X +80 = 0
FACTOR THE ABOVE EXPRESSION, OBTAINING
(3X +10)(1X +8) = 0
THEN 3X = -10 AND 1X = -8
DIVIDING TO SOLVE, X = -3 1/3 OR X = -8
```

4X**2 -7X  = 3X**2 -4X -2
YOUR ANSWERS ARE X = -10, X = -1

YOU HAVE MADE AN ERROR.
FIRST COLLECT TERMS BY SUBTRACTING 3X**2, -4X, AND -2 FROM BOTH SIDES. THIS RESULTS IN
X**2 -3X +2 = 0
TO FACTOR THE ABOVE EXPRESSION, REMEMBER THAT THE PRODUCT (PX+Q)(RX+S) IS A SIMILAR TRINOMIAL.
COMPARE THE TWO.

$$(PX+Q)(RX+S) = (PR)X^{**}2+(QR+PS)X+QS$$
$$X^{**}2 -3X +2$$

THEN QR + PS = -3
Q * S = 2
Q*S IS POSITIVE, SO Q AND S HAVE THE SAME SIGNS. SINCE THE LINEAR TERM IS NEGATIVE,
Q AND S MUST ALSO BOTH BE NEGATIVE.
CONSIDERATION OF THE POSSIBLE FACTORS OF 2 AND 1 SHOWS THAT ONLY (X-2)(X-1) GIVES THE
PROPER PRODUCT.
THEN X = 2 AND X = 1

Figure 3. Changing explanations.

```
HELLO. LET≠S DO SOME ALGEBRA PROBLEMS. FIRST TYPE YOUR NAME ON THE TYPEWRITER.
*FACTOR      ≠ YOUR ANSWERS ARE   ≠
MAKNUL(C) ≠X ≠ SIGN(D) ≠)≠ // ≠SET EACH FACTOR EQUAL TO ZERO.≠  //
MAKNUL(C) ≠X ≠ SIGN(D) ≠ = 0≠  //  MAKNUL(A) ≠X ≠ SIGN(B) ≠ = 0≠  //
≠SOLVE THE RESULTING LINEAR EQUATION.≠ // ≠X = ≠ SIDE1 ≠X = ≠SIDE2
****
```

```
FACTOR1

X**2 +3X -10 = 0
YOUR ANSWERS ARE X = 1, X = 5
FACTOR THE LEFT MEMBER INTO (X +5)(X -2)
SET EACH FACTOR EQUAL TO ZERO.
X -2 = 0
X +5 = 0
SOLVE THE RESULTING LINEAR EQUATION.
X = -5X = 2
```

```
HELLO. LET≠S DO SOME ALGEBRA PROBLEMS. FIRST TYPE YOUR NAME ON THE TYPEWRITER.
*MULTILIN      ≠ COLLECT ALL TERMS CONTAINING X≠ QUOTE ≠S ON THE RIGHT. THIS IS ≠
//  OUT] ≠ = ≠ MAKNUL(C) ≠X ≠ SIGN(D) //  ≠THEN DO THE SAME ON THE LEFT. ≠ //
MAKNUL(A) ≠X ≠ SIGN(B) ≠ = ≠ MAKNUL(C) ≠X ≠ SIGN(D) //
****
```

-4X -2 -4X +5 = 2X -3 -3X +3
YOUR ANSWER IS 2/5

YOU HAVE MADE AN ERROR. THE PROBLEM IS
-4X -2 -4X +5 = 2X -3 -3X +3
COLLECT ALL TERMS CONTAINING X≠S ON THE RIGHT. THIS IS
-4X -2 -4X +5 = -X
THEN DO THE SAME ON THE LEFT.
-8X +3 = -X

SUBTRACT -1X FROM BOTH SIDES. THIS PRODUCES
-7X +3 = 0
SUBTRACT 3 FROM BOTH SIDES, OBTAINING
-7X = -3
NOW DIVIDE BOTH SIDES BY -7, AND X = 3/7

Figure 4. Record of results printed out on the teacher's demand. The program prints out the records kept for all students who have used it, not just the one shown here.

RESULTS
STUDENT
FACTOR1 4/5
FACTOR1 5/5
FACTOR 4/5
FACTOR 4/5
FACTOR 4/5
FACTOR 4/5
SIMPLIFY1 4/5
SIMPLIFY1 4/5
SIMPLIFY 5/5
SIMPLIFY 4/5
SIMPLIFY 4/5
SIMPLIFY 4/5
SQ1 4/5
SQ1 4/5
SQ 4/5
SQ 4/5