

Computer Science Honours Project.

by David Benseman

October 1983.

The Design and Implementation of a
Linear Equation Teaching Package
for the Apple Computer.

Acknowledgements

I would like to acknowledge the help of the following people :

Mr. G. Findlay of Riccarton High School for his assistance
and criticism, and for giving me the idea in the first place.

Mr. Ellwood of Burnside High School for his
assistance in testing the package on his students.

Mr. M. Wall of Christchurch Teachers College for
his assistance in programming problems on the Apple
computers, giving me access to the schools computers,
and for providing additional software and texts.

Contents

[1]. Introduction	1
[1.1] Reasons for choosing this project.	
[1.2] Decision made on research project.	
[1.3] Transportable and easy to use.	
[1.4] Main users.	
[1.5] Records kept.	
[1.6] Teachers section.	
[2]. Other Teaching Systems	3
[2.1] Supermath.	
[2.2] Pilot.	
[2.3] Poly.	
[2.4] Logo.	
[2.5] What design goals implemented.	
[3]. Philosophy of Approach	8
[3.1] What to implement the package on.	
[3.2] Security.	
[3.3] The need for better graphics.	
[3.4] Mastery levels.	
[3.5] Breaking down of problems.	
[3.6] Examples.	
[3.7] Record file structure.	
[3.8] Teachers section.	
[3.9] Buzzer.	
[4]. What has been Achieved	14
[4.1] Programs written.	
[4.2] Testing.	
[4.3] Teachers opinions.	
[4.4] What the students thought of it.	
[5]. How the Programs Works	17
[5.1] Booting the Apple.	
[5.2] Restricting Access.	
[5.3] Errors.	
[5.4] Communicating with students.	
[5.5] Student access method	
[5.6] Student been deleted.	
[5.7] Teachers section.	
[6]. References Used.	25
[7]. Appendices	
[A]. Program listing for access program.	
[B]. Program listing for teaching package.	
[C]. Sample screen output.	
[D]. Teacher Documentation.	
[E]. Student Documentation.	

[1]. Introduction.

1.1 Reasons for choosing this project.

Having chosen a career of teaching maths at secondary school level, it was decided to undertake a research project which would be relevant to this, and also have some practical value. Thus, I approached the head of the maths departments at both Riccarton High School and Burnside High School in Christchurch asking for their ideas for a research project topic. They were both of the opinion that there were several students each year entering the sixth form with little knowledge of basic algebraic manipulation as taught at fifth form level. This included such problems as linear and quadratic equations, and simplifying equations. This greatly handicapped the student as this was assumed knowledge, and a basis for much of the sixth form syllabus.

1.2 Decision made on Research project.

It was therefore decided to write a package which would attempt to teach the students how to solve linear equations, giving both problems and examples, and if time was available, to write a program which attempt to teach students how to solve quadratic equations. A method would also be required, within each of these programs, to decide what type of problem to give to the student.

1.3 Transportable and easy to use.

Because this package was to be designed for use in schools, it was necessary to write the package in a way that was easily transportable to as many schools as possible, and to be implemented with the minimum of effort by the teacher.

1.4 Main users.

As said before, the package is aimed at sixth form students who are having problems with linear equations, but it could also be used by fifth form students who want extra work on the syllabus, as the package is based on the fifth form syllabus.

1.5 Records kept.

The package would also need to keep records of the progress of each student, thus allowing the teacher to gauge progress for each student, and see if extra help is needed.

1.6 Teachers section.

It would be necessary to write a program which would allow the teacher access to these records, thus giving him the ability to see the progress of each student. This section would need to be easy to use, and contain all the commands which would be necessary to let the teacher find out how well each student is progressing.

[2]. Other Systems Available.

2.1 Supermath

Two programs, both called Supermath, were looked at when looking at the design of the package. These programs both used a system of levels where, when a student gets a problem correct, they go up a number of levels, depending on how many mistakes they make while doing the problem. If they get the problem wrong, they go down a number of levels. The two programs differ greatly in complexity, but not by much in what they teach. One program is written by Ron Graff of Apple computers and teaches addition, subtraction, multiplication, and division using the low resolution graphics screen on the Apple. Its numbers are drawn using straight lines, giving digits similar to those on a calculator display. The other, a more complex program, uses high resolution graphics to display addition, subtraction, long division and long multiplication problems, giving some positive reinforcement with pictures which move if the student gets the problem correct. For example, one picture is an octopus which moves all its tentacles, by showing a sequence of four pictures of it in different positions.

2.2.1 Pilot.

One package which most schools have is Pilot, which allows the teacher to write a series of problems, and a list of possible answers to each problem. These are programmed into the computer in an easy to use high level language.

Some implementations also keep records of how well each student answered the questions.

2.2.2 Disadvantages of Pilot.

This package, although widely available, is not used much due to the amount of time needed in programming each lesson on the computer. This approach does not give any examples, or give the ability to do some problems and then go away and come back later to finish them off.

2.3.1 Poly Computers.

A system which was developed in New Zealand with the main aim of being used for teaching purposes is the Poly computer system. Many teaching packages have been written for this system, including one package which 'teaches' linear equations. This program generates a problem and then displays it to the screen. The student is then asked to enter the answer in the simplest fraction possible. Eg.-3 7/9, not -34/9, -3 14/18 etc. This package has five levels of difficulty and to advance to a higher level, it is necessary to answer five problems in a row correctly.

2.3.2 Disadvantages of Poly.

These problems become quite hard quite quickly, with level three asking problems like $-6(-4x+4)-7(-7x-5)=-5(9x-6)$, and expecting the answer with no intermediate working. This can be thought of as a testing program, rather than a teaching package, as it is necessary to know how to solve linear equations in order to answer any problems. This

program does not give you any examples on how to do linear equations in order to go about solving any problems. There is also no way of stopping the computer asking more problems without advancing to level five and answering five questions in a row at this level, other than typing the EXIT key which stops the program, and returns to the operation level, with no way for a user who does not know how to use the Poly to restart the program other than rebooting the disk.

2.3.3 An improvement with the quadratic equations.

The quadratic equation program was slightly better in that you could specify the number of problems you wished to do. The problems were also found to be easier, with a variety of ways of presenting each problem. For example, problems were posed like : $x^2=10x-16$, $x^2-10x+16=0$, or $(x-2)(x-8)=0$, and ask for one root then the other, which could be entered in either order.

2.4 Logo.

This is another package available for the Apple computers, but unlike Pilot, is used by many schools as a teaching language. This language's main drawcard is its graphics use, giving the user the ability to draw shapes using a 'mouse', which is controlled by commands like 'LEFT 90', 'FORWARD 30', etc. This language is easy to use, and interpretive, allowing the student to see immediate results to each command. The language is also recursive, allowing the user the ability to create commands like: to climb ladder, if not at the top, move up one rung, climb ladder,

using primitives and building on each of these.

2.4 What design goals were implemented.

2.4.1 The idea of levels.

It was decided to use the levels approach in giving problems to the student, as used by Supermath. This allowed the teacher to find out how well a student is progressing by relating the number of problems done, the level the student is at, and the number of problems answered correctly. It also allowed a simple way to give problems which are relative in difficulty to the level the student is at.

2.4.2 Examples.

So as the student knows what is required of him, an example would need to be displayed at each level. This is also done to teach the student a method of solving that type of equation.

2.4.3 Random problems.

So the package does not run out of problems to give the student, it was decided to produce random numbers for each problem. This first selects a value for 'x', then randomly selects a value for all but one of the other variables in the equation, the last variable being chosen by putting the other values into the equation. These numbers could not be completely random as that could produce both errors and trivial problems. An example of an error which could occur is in $ax+b=cx+d$, where $a=c$, thus producing an equation of the

form $0x=0$ which would make the program crash when dividing by zero. Also the solution to $0/0$ is meaningless in this context. An example of a trivial problem would be something like $1x$, or $ax+0$, where it would not be necessary to perform any steps to solve the problem.

[3]. Philosophy of approach.

3.1 What to implement the package on.

As most New Zealand schools now have Apple computers, due to the Apple computer company selling them cheap to educational institutions, it was decided to write the package for the Apple computers, storing the program on floppy disk. This approach means that the teacher needs only to give the student the disk containing the package to use, and have the program teach the student how to solve linear equations without further intervention from the teacher.

3.2 Security.

As many of the students have had some teaching in using the Basic language, it would not be hard to write a program which would access their records and alter them to show that they were doing better than they were. Because of this, control characters were embedded in the name of the records file which cannot be seen on the screen, but without which, nothing can be done to the file, as the message 'FILE NOT FOUND' will be displayed on the screen.

3.3 The need for better graphics.

Many Apple computers owned by schools do not have a lower case character chip, meaning that everything has to be written in upper case. This gives the impression to the student that the computer is shouting at the him, which is not good teaching practice.

3.3.1 Higher Text graphics generator.

Because of the above problem, a graphics package called 'HIGHER TEXT' was used, which takes output to the text screen, and plots this in the high resolution graphics screen. This gives characters twice as high and twice as wide as the normal characters. These characters look much more 'friendly' to the user, and also allows a mixture of both upper and lower case characters to be used. The large size also helps students with reading difficulties. The fact that Higher Text uses the high resolution graphics screen was also used to draw horizontal lines for use in division.

3.3.2 Disadvantages of Higher Text.

There are some disadvantages of Higher Text in that it takes up a lot of space, restricting the size of the program. Higher Text takes up a lot space due to the program taking up some of the area set aside for programs, and the fact that it uses the high resolution graphics screen, which stores each pixel in two bits, corresponding to the colour of that pixel. Because the generator, the fonts (the designs for each character) and the high resolution graphics screen are in areas of memory which can be used for programs, this effectively halves the size of the program capable of being written for the Apple. This has meant that several statements have had to be written on each line as a space saving measure. The reason that this reduces the size of the file, is that each new statement label in a program takes up extra space. This also serves the purpose of speeding up the

execution of the program. The other space saving measure used was to keep comments to a minimum. Another problem with Higher Text is that it only allows twenty characters per line and ten lines per screen, meaning that the organization of the screen was made more restrictive.

3.4 Mastery levels.

A system of levels was used for several reasons. It means that the student has a goal of level 99 to achieve, which is the highest level. To get to this level, the student must get at least four problems at each level range correct, starting from his initial level, more if they make any mistakes. It also allows the teacher to gauge progress by seeing what level the student is at. The other reason for the levels is that it gives the chance to give the student problems which are relative in complexity to their level of mastery. The sort of problems given at each level are :

Level	Problem	Types
=====	=====	=====
0 - 9	$ax = b$	all natural numbers.
10-19	$ax = b$	all integer.
20-29	$a/b \ x = c$	all integer.
30-39	$ax = b$	x integer. a,b real.
40-49	$x + a = b$	all integer.
50-59	$ax + b = c$	all integer.
60-79	$ax + b = cx + d$	all integer.
80-99	$a(bx + c) = c$	all integer.

3.5 Breaking down of problems.

As each problem is solved, it is broken down into simpler problems. Eg. a problem in the range 80-99 is first divided by a , giving $bx+c=(d/a)$, which is the same form as those in the range 60-79. This is then reduced to $bx=(d/a-c)$ which is the same as those in the range 10-19. This was done to show the student that a problem can be broken down into smaller problems, which are easier to solve. This means that when they try to solve any problems outside the scope of this package, they can simplify the problem and solve it.

3.6 Examples.

So the student knows what input the computer expects in order to solve the problem, examples are given at all levels. An example is given at the start of each session if the student wants it. If they do not want the example, they can type 'Q', which skips the example and goes straight to the problems. Another way of getting an example is to type 'H' for help, when asked for any input while solving a problem. This gives an example (not the problem being given when 'H' was typed) and then restarts the problem. The student can see the whole problem from start to finish, or he can type 'CTRL C' during a pause to start another problem.

3.7 Record file structure.

The records file is a random access file which must be on disk. Each record is 64 bytes long, and is treated like a text file by the apple disk operating system, meaning that

each field of a record is a line of characters. Each record is therefore access by the starting position on the file and adding the length of the record times the record number required. This accessing is done by the disk operating system, not by the users program. The structure of the file is :

Record 0 :

Teachers password.

Number of students using the package.

Number of the record at the head of the linked list of free records.

The number of the student who last used the package.

This was used in communicating between the access and teaching programs, as all variables are lost when the other program is run.

All other records :

A number to say if the student has been deleted or not.

A zero means the student has been deleted, a one means that the student is still on the records.

If this number is zero, the only other field in this record is the link to the next free record.

The name of the student the record belongs to.

The level the student is at in linear equations.

The number of linear equation problems done.

The number of linear equation problems correct.

The level the student is at in quadratic equations.

The number of quadratic equation problems correct.

The number of quadratic equation problems correct.

3.8 Teachers section.

The teachers section envisaged in section 1.6 was decided to be programmed into the same program as that allowing students access, putting the code at the end of the student access section. This allows the same security to be used for both student and teacher sections.

3.9 Buzzer.

When a student makes a mistake while doing a problem, it was decided to write on the screen that they had made a mistake, as well as give some audible indication of this. Unfortunately, Higher Text had redesigned the 'CTRL G', or bell, to a pleasant, and somewhat cheerful sound, meaning that the computer sounded happy at the student answering a problem incorrectly. Because of this, it was decided to use a command of the Applesoft language which allowed the speaker to be 'toggled' once. This command was put into a tight FOR loop which toggled the speaker several times, altering the tone to a somewhat less cheerful note.

[4]. What has been achieved.

4.1 Programs Written.

To date, two programs have been written for this project, with another program intended to be written next year. One program allows the student on to the teaching package, checking that the student number and names are the same on the disk as those entered. This ensures that the correct records are updated when the student runs the package. The other program generates random numbers for both problems and examples, and presents these to the student in a way which is aimed at teaching the student how to do linear equations, without the help of the teacher.

4.2 Testing.

The program was extensively tested while the programs were being developed. This managed to find most of the bugs, but not all. Later, as part of my testing, the disks were lent to teachers at both Riccarton and Burnside High Schools for their comments. They were also asked to test them on some of their pupils. This was done, and both schools managed to find bugs in the program. These bugs were mainly minor screen details which were easily fixed. For example, the word 'character', which is frequently used by people who use computers, was found to be not understood by many students, and had to be replaced by the word 'letter' throughout the program and student documentation.

4.3 Teachers opinions.

Teachers at Burnside High School were pleased with the package, saying that the approach to the problems were good, with a good layout and sequence of steps. They were particularly taken with the large characters which were provided by Higher Text, saying that they were easy to read, and therefore conveyed the message clearly. They also said that they would be willing to use the package at their school, but would prefer it to be written for their BBC micro computers. The teacher at Riccarton High School was unable to have any of his pupils test out the program, but he tested the program with the specific aim of finding any faults. He found a few faults in the layout, all of which were corrected. The package as a whole he found to be good, and said he would, if given the chance, use the package for teaching purposes.

4.4 What the students thought of it.

This is the main test of the package, does it actually teach the student anything, and can they understand from the package what they are supposed to do. The students at Burnside High School the program was tested on did not have a chance to have a look at the documentation telling them what was required at each step, but found little difficulty in understanding what they had to type in. They managed to find a few bugs in the program, which have rectified. Some of their comments were taken into consideration and parts of the program modified to accommodate their suggestions. One

suggestion they had was that some of the pauses in the examples were too long when an example was given, so this particular length pause was shortened. They were all of the opinion that the package actually taught them something, and was easy to use, once the basics of how to use the computer was mastered. They were even willing to use the package into the lunch hour before having to go to lunch. One student said that he found it fun to use the package, which is always a help when trying to teach a student anything. Another student asked if I would be coming back to try it out on them again, saying that he would be willing to have another go at it.

[5]. How the program works.

5.1 Booting the Apple.

When the Apple computer is switched on, the program which the disk was initialized with, is loaded and run. This package has the disk initialized with a program which allows the student access to the teaching program, and the teacher access to the records to see how each student is progressing.

5.2 Restricting Access.

The access to the records has had to be restricted to the programs I have written, and not to the user. This is to prevent corruption of the records by either putting in invalid data, or by a student changing the records to make it look as if he is doing better than he is. This is done by control characters embedded in the file name of the records, which can not be seen on the screen.

5.3.1 Errors

To prevent the student from crashing the program, an 'ONERR GOTO 1000' statement was inserted at the start of the program. This means that any error that occurs during the execution of the program which would normally make the operating system give an error message and stop are handled by the routine starting at line 1000.

5.3.2 Interrupt attempt handled.

This routine handles errors like a 'CTRL C' interrupt attempt, displaying the message 'DON'T DO THAT'. The errors handled relate to attempts by the user to crash the program; not errors in the program, which are handled by printing the error and stopping the execution of the program.

5.3.3 Input Errors.

All student input is read in as a string, and then converted to a number with the Applesoft function VAL to prevent students from entering in characters instead of numbers, but not so for the teachers. This is because teachers are assumed to know something about what they are doing, whereas the students are assumed to not be able to use the computer very well, other than to play games. Because of this, when a 'Bad response to input' error occurs during execution of the program, the message 'IT WAS ASSUMED THAT TEACHERS WOULD TYPE NUMBERS WHEN ASKED FOR NUMERICAL INPUT!' is displayed. When this error and the CTRL C interrupt attempt error occur, the message is displayed for about a second, and then the cursor is moved back to the line it came from and the message is deleted from the screen.

5.3.4 I/O Errors.

If an 'out of data' error or 'disk I/O error' occurs, then the program tries to place which statement the error occurred in, and then give an error message which is appropriate to where it happened. If the error occurred when

a student was trying to get onto the system, it tells the student that there is an error in his data and then gives him another number to carry on with and do some problems. If it is in the teachers section of the program, it gives the error message 'DATA ERROR IN STUDENT ' and gives the number of the student. It then branches to the part of the program appropriate to where the error came from.

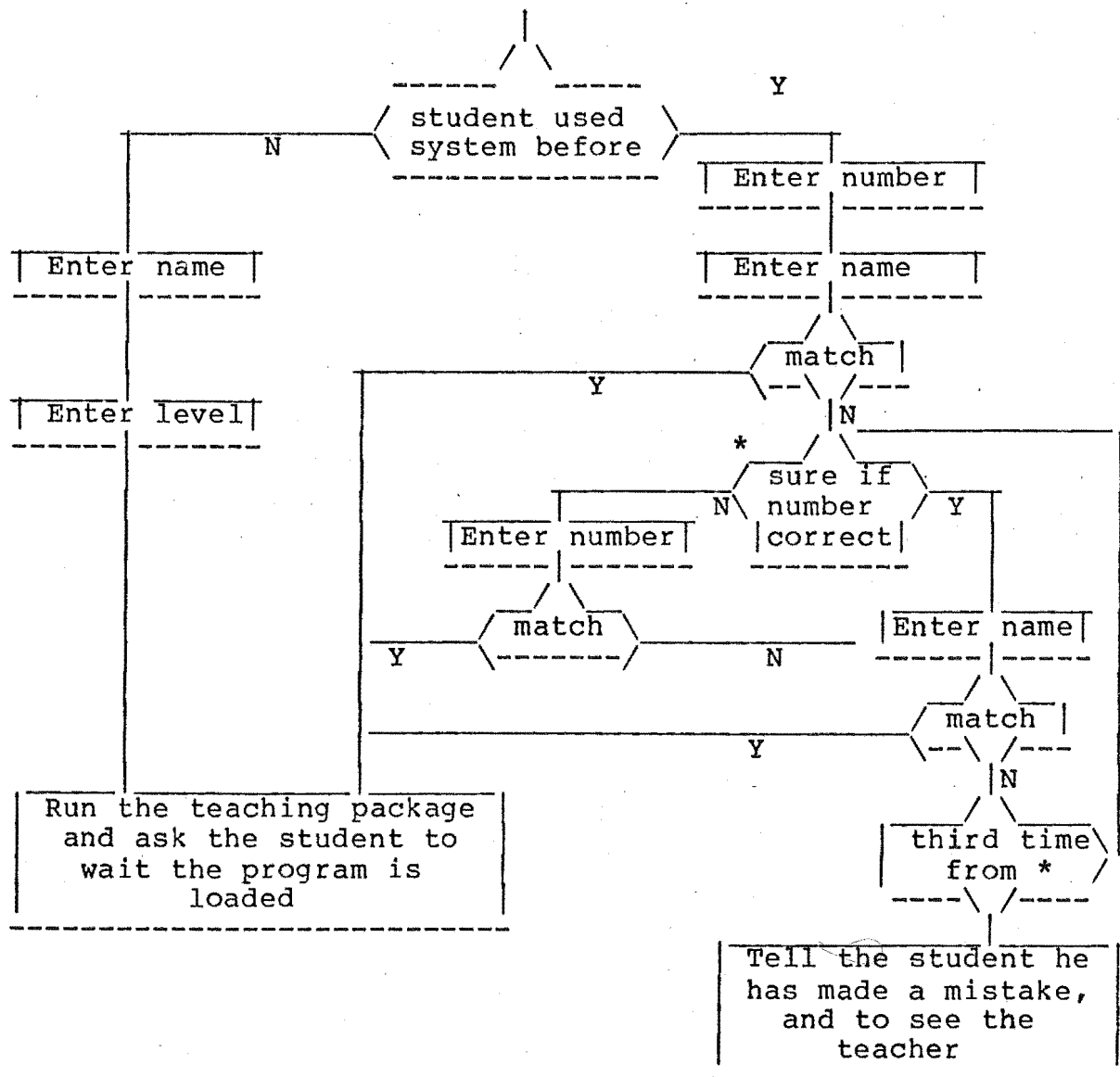
5.3.5 File not found error.

If it was in the part of the program where the other program is run from, then it gives the message that the file is not present, and could the student insert the correct disk with the file LINEAR EQUATIONS on it, asking the student to type in a letter once they have done so.

5.4 Communicating with students.

The word letter, rather than character was used when communicating to the student in all parts of the program. This is because most students are not sure what a character is, whereas they know what a letter is.

5.5 Access to teaching package.



Flow chart showing sequence of steps for a student to get onto the package.

5.6 Student been deleted.

If the student tries to use the system once his record has been deleted, the program tells the student that his records have been deleted, and asks him to see the teacher about it.

5.7 Teacher's section.

To use the teacher's section, the teacher types in at the start that he has used the program before, and that his student number is 0. He is then asked for his password, which is not echoed to the screen as he type it in. Once he has entered the correct password, as stored in record zero of the records file, a menu of what can be done is displayed on the screen. The options that are available are :

1). Display all student records.

This displays all student records, with two lines per student, one for the name and one for the information on that student. The information is displayed one screen at a time, each screen containing up to eight student records. At the end of each screen, the option is given to carry on with another screen, or to return to the main menu. The option is also given at the start to have a hard copy printout, if a printer is connected.

2). Display progress for a range of student numbers.

This is operationally the same as 1) except that it allows the teacher to just display the records for a range of student numbers. This is useful if there are a

large number of students using the package and the teacher wanted to see the records for those students who had recently started using the package, and therefore had large student numbers. This first asks for the lower bound, and then the upper bound of student numbers which progress is required, then carries on the same as 1).

3). Display records for individual students.

This command goes into a loop, asking for a student number for which data is required, the loop being terminated by '0' being typed in as the student number. If there are no error messages come up at the screen, then the progress of the student asked for is displayed.

4). Delete individual student records.

This goes into a loop, similar to that in 3), asking for numbers of students to delete, terminated by a '0' being typed in at the terminal. This deletes students, putting the number of the student being deleted in record zero of the records file, and putting the old number contained in record zero in the record being deleted and marking that record deleted. This therefore sets up a linked list of free spaces in the file, inserting new nodes at the start of the list. When students who have not used the package before use the package, they are allocated these numbers in the free list before being allocated new numbers larger than those present.

5). Delete all student records.

If there are a large number of students using the package, space will be allocated on the disk for all these records. If, say at the end of the year, the teacher wishes to delete all the records to make room on the disk for more students next year, this option will first delete the records file from the disk, therefore freeing the disk space, and then recreating the file by reinitializing record zero of the file. To prevent accidental erasure of the file, the teacher is asked if he is sure that he wishes to delete all student records.

6). Find out how many students are using the system.

This option writes the number which is stored in record as the maximum student number, and tells the teacher that there may be less than this if the teacher has deleted any. This option was given so the teacher can find out the number, and use it in options two and four.

7). Change the password.

As the package is initialized with the password as TEACHER, and this is written in the documentation, an option to change the password was used. This gives the teacher the option of having what he is typing in echoed to the terminal for verification, or hidden, like when he typed in his password to get onto the package. Next time the teacher uses the system, he will need to use this new password.

8). Exit.

This option exits from the program, and into Applesoft. From here the teacher can list the program to see how it works, and maybe fix any bugs if they occur.

[6] References used.

1). Form 5 Mathematics.

- Ellwood and Guerin, Burnside High School.

The problems were based on the syllabus contained in this text book, with the types of problems given in the book divided up into levels and randomly generated in the program. This text book was written by teachers at Burnside High School, and is also used at other schools.

2). Higher Text.

- Darrel and Ron Aldrich.

This is the users manual for the Higher Text graphics generator package, used by the teaching program.

3). Program Line Editor.

- Neil Konzen

A users manual for the program line editor which was used when writing the programs. A piece of software, which considering its usefulness, should be in ROM, rather than having to load it every time you boot the disk.

APPENDIX A.

Program listing for access program.

Access Program for Linear Equation Teaching Package.

```
70      ONERR GOTO 1000
      : XX = 0
80      REM   DUE TO SPACE LIMITATIONS ON THE APPLE COMPUTER, A BAD PR
      OGRAMMING STYLE OF SEVERAL STATEMENTS ON ONE LINE HAS HAD
      TO BE ADOPTED FOR THIS PROJECT.
90      POKE 33,40
      : POKE 32,0
      : TEXT
      : POKE - 16384,0
      : NOTRACE
      : PRINT CHR$(4)"NOMON C,I,0"
100     HOME
      : D$ = CHR$(4)
      : FIL$ = " RE[ ]C(cE)OR(cU)R(cH)DS"
      : OPN$ = D$ + "OPEN" + FIL$ + ",L64"
      : RDS$ = D$ + "READ" + FIL$ + ",R"
      : WRT$ = D$ + "WRITE" + FIL$ + ",R"
      : CLS$ = D$ + "CLOSE" + FIL$
      : DLT$ = D$ + "DELETE" + FIL$
120     PRINT "          BASIC ALGEBRA."
      : PRINT "          ====="
      : PRINT
140     PRINT "  LINEAR EQUATIONS."
150     PRINT "  QUADRATIC EQUATIONS."
160     VTAB 7
      : PRINT "HAVE YOU USED THIS PACKAGE BEFORE ?"
      : CALL - 958
      : INPUT "(Y/N) : ";ANS$
170     ANS$ = LEFT$(ANS$,1)
      : IF ANS$ = "Y" THEN 350
180     IF ANS$ < > "N" THEN PRINT "(cG)"
      : GOTO 160
190     PRINT OPN$
200     PRINT RDS$;0
210     INPUT N,NAME$,AVAIL
      : IF AVAIL = 0 THEN 250
220     PRINT RDS$;AVAIL
      : INPUT N1,N2
      : NUM = AVAIL
      : AVAIL = N2
230     PRINT WRT$;0
      : PRINT N
      : PRINT NAME$
      : PRINT AVAIL
      : PRINT CLS$
240     GOTO 260
250     N = N + 1
      : NUM = N
      : GOTO 230
260     VTAB 10
```

```

      : CALL - 868
      : PRINT "WHAT IS YOUR NAME"
      : PRINT "(FIRSTNAME SURNAME)"
270   VTAB 12
      : CALL - 868
      : INPUT ": ";NAME$
      : IF LEN (NAME$) > 30 THEN 300
280   CALL - 958
      : PRINT "YOUR STUDENT NUMBER IS ";NUM
      : PRINT "PLEASE REMEMBER THIS NUMBER AS YOU WILL NEED IT TO USE
          THE PACKAGE AGAIN."
290   GOTO 700
300   VTAB 13
      : CALL - 958
      : PRINT "DUE TO SPACE LIMITATIONS, THE LENGTH OF NAMES MUST BE
          LESS THAN 30 CHARACTERS."
310   GOTO 260
350   REM      STUDENT USED THE SYSTEM BEFORE
360   VTAB 10
      : CALL - 868
      : INPUT "WHAT IS YOUR STUDENT NUMBER : ";NUM$
      : NUM = VAL (NUM$)
370   IF NUM = 0 AND NUM$ = "0" THEN 30000
380   IF NUM = 0 THEN PRINT "(cG)INVALID INPUT"
      : GOTO 350
385   GOSUB 410
      : IF ERR THEN 350
390   CALL - 868
      : VTAB 12
      : PRINT "WHAT IS YOUR NAME ?"
      : PRINT "(FIRSTNAME SURNAME)"
      : CALL - 868
      : INPUT ": ";NAME$
400   NAME$ = LEFT$ (NAME$,30)
      : GOTO 450
410   IF NUM < 1 THEN PRINT "NUMBER TOO SMALL"
      : ERR = 1
      : RETURN
415   PRINT OPN$
      : PRINT RDS$;0
420   INPUT N
      : IF N < NUM THEN PRINT "NUMBER TOO LARGE!"
      : PRINT CLS$
      : ERR = 1
      : RETURN
425   PRINT RDS$;NUM
430   INPUT X$
      : IF X$ = "0" THEN PRINT "YOU HAVE BEEN DELETED FROM THE FILES!
          PLEASE SEE YOUR TEACHER"
      : PRINT CLS$
      : POP
      : END

```

```

435 INPUT NME$,LLEVEL,LDNE,LCORRECT,QLEVEL,QDNE,QCORRECT
440 PRINT CLS$
   : ERR = 0
   : RETURN
450 IF NAME$ = NME$ THEN 2000
455 SW = 0
460 VTAB 15
   : CALL - 958
   : PRINT "DATA DISSAGREES WITH RECORDS FOR STUDENTNUMBER ";NUM
470 PRINT "ARE YOU SURE YOU HAVE THE RIGHT NUMBER?"
   : CALL - 868
   : INPUT "(Y/N) : ";ANS$
   : IF "Y" = LEFT$(ANS$,1) THEN 530
480 IF "N" < > LEFT$(ANS$,1) THEN 460
490 INPUT "RE-ENTER YOUR NUMBER : ";NUM$
   : NUM = VAL(NUM$)
500 GOSUB 410
510 IF ERR AND SW < 2 THEN 560
515 IF ERR THEN 570
520 IF NAME$ = NME$ THEN 2000
530 PRINT "RE-ENTER YOUR NAME : "
   : INPUT " : ";NAME$
540 IF NAME$ = NME$ THEN 2000
550 IF SW = 2 THEN 570
560 SW = SW + 1
   : GOTO 460
570 PRINT "ARE YOU SURE YOU HAVE THE CORRECT DISK WITH YOUR RECORDS
      ON IN THE SLOT ?"
   : INPUT "(Y/N) : ";ANS$
580 ANS$ = LEFT$(ANS$,1)
   : IF ANS$ = "Y" THEN 595
585 IF ANS$ < > "N" THEN 570
590 PRINT "PLEASE NOW INSERT THE CORRECT DISK AND I WILL START AGAIN
      FROM THE BEGINNING."
   : FOR I = 1 TO 5000
   : NEXT I
   : GOTO 110
595 PRINT "YOU HAVE MADE A BOO BOO"
   : PRINT "EITHER YOU HAVE YOUR NUMBER OR THE NAME YOU USED LAST
      TIME WRONG. GO AND SEE YOUR TEACHER AND HE WILL FIND OUT YOUR
      NUMBER. TO RE-RUN THIS PACKAGE, REBOOT THE DISK."
   : END
700 VTAB 24
   : PRINT "PRESS A LETTER TO CONTINUE : ";
   : GET ANS$
   : HOME
   : LIN = 1
   : GOTO 780
   : PRINT "WHICH SET OF PROBLEMS DO YOU WISH TO DO?"

```

```

: PRINT
710 PRINT
: PRINT "      1.LINEAR EQUATIONS."
720 PRINT
: PRINT "      2.QUADRATIC EQUATIONS."
730 VTAB 10
: CALL - 958
: INPUT "TYPE NUMBER OF YOUR CHOICE : ";ANS$
: IF LEN (ANS$) < > 1 THEN 730
740 IF ANS$ = "1" THEN LIN = 1
: GOTO 780
760 IF ANS$ = "2" THEN PRINT "NOT IMPLEMENTED YET "
: FOR I = 1 TO 1000
: NEXT I
: GOTO 730
770 GOTO 730
780 VTAB 12
: PRINT "WHAT LEVEL DO YOU WISH TO START AT.      0 IS THE EASIE
      ST,99 IS THE HARDEST.      REMEMBER, THE HARDER YOU START Y
      OURSELF OFF AT, THE LESS LIKELY YOU ARE OF      GETTING TH
      E ANSWERS CORRECT."
790 CALL - 868
: INPUT "LEVEL : ";ANS$
: IF LEN (ANS$) > 2 THEN 780
800 L = VAL (ANS$)
: IF L < 0 THEN 780
820 PRINT OPN$
: PRINT WRT$;NUM
: PRINT 1
: PRINT NAME$
: PRINT L
: PRINT 0
: PRINT 0
: PRINT L
: PRINT 0
: PRINT 0
: PRINT RDS$;0
: INPUT A$,B$,C$
: PRINT WRT$;0
: PRINT A$
: PRINT B$
: PRINT C$
: PRINT NUM
: PRINT CLS$
830 HOME
: PRINT "R E M E M B E R !"
: PRINT "WHEN YOU WISH TO USE THE PROGRAM AGAIN, USE:"
: PRINT "NAME      = ";NAME$
: PRINT "NUMBER = ";NUM
840 IF LIN = 1 THEN PRINT D$;"RUN LINEAR EQUATIONS"
850 VTAB 10
: PRINT "PLEASE WAIT."
1000 ZZ = PEEK (218) + PEEK (219) * 256
: YY = PEEK (222)
: V = PEEK (37)
: IF (XX = 176 OR XX = 254 OR XX = 255) THEN RETURN

```



```

1010  XX = YY
      : IF YY = 176 THEN 1050
      : REM      INPUT STRING TOO LONG.
1020  IF YY = 254 THEN 1060
      : REM      BAD RESPONSE TO INPUT.(NUMERICAL)
1030  IF YY = 255 THEN 1070
      : REM      CTRL C INTERRUPT ATTEMPTED.
1035  IF YY = 42 OR YY = 5 THEN GOTO 1100
1050  POKE 216,0
      : PRINT "PROGRAM ERROR "YY"IN LINE "ZZ
      : PRINT "PLEASE INFORM THE TEACHER"
      : END
      : REM      AN ERROR IN THE PROGRAM.
1055  PRINT "INPUT STRING TOO LONG. PLEASE REFRAIN FROM PLAYING!"
      : GOTO 1080
1060  PRINT "IT WAS ASSUMED THAT TEACHERS WOULD TYPE NUMBERS WHEN AS
      KED FOR NUMERICAL INPUT!"
      : GOTO 1080
1070  PRINT "DON'T DO THAT!"
1080  FOR J = 1 TO 1500
      : NEXT J
      : VTAB V
      : CALL - 958
      : XX = 0
      : RESUME
1100  IF ZZ < 1000 THEN PRINT "THERE IS AN ERROR IN YOUR DATA.
      CAN YOU SEE YOUR TEACHER AND ASK FOR      THIS RECORD TO
      BE DELETED."
      : PRINT "IN THE MEAN TIME,I WILL LET YOU CARRY ONWITH A NEW NUM
      BER."
      : PRINT "PRESS A LETTER TO CONTINUE : ";
1105  GET ANS$
      : GOTO 190
1110  PRINT "DATA ERROR IN STUDENT ";I
      : IF ZZ = 30240 OR ZZ = 30250 THEN LOWER = I + 1
      : IF LOWER < = UPPER THEN 30235
1115  GOTO 30305
1120  IF ZZ > = 30610 THEN 30640
1130  GOTO 1040
2000  ANS$ = "1"
      : GOTO 2040
      : HOME
      : PRINT "WHICH SET OF PROBLEMS DO YOU WISH TO DO?"
2010  PRINT
      : PRINT "      1.LINEAR EQUATIONS."
2020  PRINT
      : PRINT "      2.QUADRATIC EQUATIONS."
2030  VTAB 10
      : CALL - 958
      : INPUT "TYPE NUMBER OF YOUR CHOICE : ";ANS$
      : IF LEN (ANS$) < > 1 THEN 2030

```

```

2040 IF ANS$ = "1" THEN PRINT "PLEASE WAIT."
: PRINT RDS$;0
: INPUT A$,B$,C$
: PRINT WRT$;0
: PRINT A$
: PRINT B$
: PRINT C$
: PRINT NUM
: PRINT CLS$
: PRINT
: PRINT D$"RUN LINEAR EQUATIONS"
: END
2050 IF ANS$ = "2" THEN PRINT "NOT IMPLEMENTED YET."
: FOR I = 1 TO 1000
: NEXT I
2060 GOTO 2030
30000 VTAB 12
: GOSUB 30010
: GOTO 30090
30010 NAME$ = ""
: COUNT = 0
: PRINT "ENTER YOUR PASSWORD : "
30020 GET CH$
30025 IF CH$ = CHR$ (13) THEN
: PRINT
: RETURN
30030 IF CH$ = CHR$ (8) THEN 30060
30040 IF CH$ = CHR$ (32) THEN PRINT "<cG>";
: GOTO 30020
30050 PRINT " ";
: NAME$ = NAME$ + CH$
: COUNT = COUNT + 1
: GOTO 30020
30060 IF COUNT = 0 THEN PRINT "<cG>";
: GOTO 30020
30070 COUNT = COUNT - 1
: IF COUNT > 0 THEN NAME$ = LEFT$ (NAME$,COUNT)
: GOTO 30080
30075 NAME$ = ""
30080 PRINT CHR$ (8);
: GOTO 30020
30090 PRINT OPN$
: PRINT RDS$;0
30110 INPUT N,NME$,AVAIL
30120 PRINT CLS$
30130 IF NAME$ < > NME$ THEN PRINT "INVALID PASSWORD"
: FOR I = 1 TO 1000
: NEXT I
: GOTO 160
30140 HOME
: PRINT "THESE ARE THE OPTIONS AVAIABLE."

```

```

30145 PRINT
      : PRINT " 1.LIST PROGRESS FOR ALL STUDENTS."
30150 PRINT
      : PRINT " 2.LIST PROGRESS FOR A RANGE OF STUDENT   NUMBERS."
30155 PRINT
      : PRINT " 3.LIST PROGRESS FOR SINGLE STUDENTS."
30160 PRINT
      : PRINT " 4.DELETE INDIVIDUAL STUDENT RECORDS."
30165 PRINT
      : PRINT " 5.DELETE ALL STUDENT RECORDS."
30170 PRINT
      : PRINT " 6.SHOW NUMBER USING PACKAGE."
30175 PRINT
      : PRINT " 7.CHANGE THE PASSWORD."
30180 PRINT
      : PRINT " 8.EXIT."
30185 VTAB 20
      : CALL - 868
      : INPUT "ENTER OPTION NUMBER : ";OPT
30190 ON OPT GOSUB 30200,30315,30350,30380,30490,30540,30560,30600
      : GOTO 30140
30195 PRINT "INVALID OPTION NUMBER."
      : GOTO 30185
30200 LOWER = 1
      : UPPER = N
30205 IF LOWER > N THEN PRINT "NO STUDENTS PRESENT WITH NUMBER > ";L
      OWER
      : FOR I = 1 TO 1000
      : NEXT I
      : RETURN
30210 PRINT "DO YOU WANT A HARD COPY PRINTOUT ?.      (ONLY IF A PRIN
      TER IS CONNECTED)"
      : INPUT "(Y/N) : ";ANS#
30215 HARD = 0
      : IF ANS# = "Y" THEN PRINT D#;"PR#1"
      : HARD = 1
30220 HOME
      : PRINT "STUDENT PROGRESS REPORT"
      : PRINT
30225 PRINT "STUDENT      LINEAR      QUADRATIC"
      : PRINT "NUMBER  LEVL ONE CRCT  LEVL ONE CRCT"
30230 SHOWN = 0
      : PRINT OPN#
30235 FOR I = LOWER TO UPPER
30240   PRINT RDS#;I
      :   INPUT X#
      :   IF X# = "0" THEN 30300
30250   INPUT NME#,LLEVEL,LONE,LORRECT,QLEVEL,QONE,QCORRECT

```

```

30260 PRINT NME$
: PRINT I; TAB( 8);"I";LLEVEL; TAB( 13);"I";LDNE; TAB( 18);"I
: ";LCORRECT; TAB( 25);"I";QLEVEL; TAB( 30);"I";QDNE; TAB(
: 35);"I";QCORRECT
30270 SHOWN = SHOWN + 1
: IF SHOWN < 8 THEN 30300
30275 PRINT CLS$
: PRINT D$"PR#0"
30280 PRINT
: PRINT "PRESS 'Q' TO RETURN TO MENU,ANY OTHER KEY FOR MORE
: ";
30285 GET ANS$
: PRINT
30290 IF ANS$ = "Q" THEN GOTO 30310
30295 IF HARD = 1 THEN PRINT D$"PR#1"
30298 SHOWN = 0
: VTAB 5
: CALL - 958
: PRINT OPN$
30300 NEXT
30305 PRINT CLS$
: PRINT D$"PR#0"
: VTAB 23
: INPUT "PRESS RETURN WHEN READY:";ANS$
30310 RETURN
30315 PRINT "ENTER RANGE OF STUDENTS YOU REQUIRE OUTPUT FOR."
30320 INPUT "LOWER BOUND : ";LOWER
: INPUT "UPPER BOUND : ";UPPER
30325 IF LOWER < 1 THEN PRINT "LOWER BOUND MUST BE > 0,NOW CHANGED T
: 0 1"
: LOWER = 1
30330 IF UPPER > N THEN PRINT "THERE ARE ONLY ";N;" STUDENTS USING"
: PRINT "THIS PACKAGE,UPPER CBOUND NOW ";N
: UPPER = N
30340 GOSUB 30205
: RETURN
30350 PRINT "WHICH STUDENT NUMBER ?"
: CALL - 868
: INPUT "TYPE '0' TO FINISH : ";I
: PRINT
30355 IF I = 0 THEN RETURN
30360 IF I > N OR I < 1 THEN PRINT "INVALID STUDENT NUMBER !"
: PRINT "MUST BE >= 1 OR <= ";N
: PRINT
: GOTO 30350
30370 GOSUB 30610
: GOTO 30350
30380 VTAB 21
: PRINT "WHAT IS THE NUMBER OF THE STUDENT YOU WISH TO DELETE
: ?"

```

```

30390 UTAB 22
      : CALL - 868
      : INPUT "TYPE '0' TO FINISH : ";I
30400 IF I = 0 THEN RETURN
30405 IF I > N OR I < 1 THEN PRINT "INVALID NUMBER !"
      : GOTO 30390
30410 PRINT OPN$
      : PRINT RDS$;0
      : INPUT N,NME$,AVAIL
30420 PRINT WRT$;I
      : PRINT 0
      : PRINT AVAIL
      : AVAIL = I
30430 PRINT WRT$;0
      : PRINT N
      : PRINT NME$
      : PRINT I
30440 PRINT CLS$
      : GOTO 30390
30490 PRINT "ARE YOU SURE YOU WISH TO DELETE ALL      STUDENT RECORDS
      ?"
      : INPUT "(Y/N) : ";ANS$
      : IF LEFT$(ANS$,1) <> "Y" THEN RETURN
30500 PRINT OPN$
      : PRINT DLT$
      : PRINT OPN$
      : PRINT WRT$;0
30510 PRINT 0
      : PRINT NAME$
      : PRINT 0
      : PRINT CLS$
30520 N = 0
      : RETURN
30540 PRINT "THERE ARE A MAXIMUM OF ";N;" STUDENTS"
      : PRINT "USING THE PACKAGE"
      : PRINT "<LESS IF YOU HAVE DELETED SOME>"
      : PRINT "PRESS ANY KEY TO  CONTINUE:";
30550 GET ANS$
      : RETURN
30560 PRINT "DO YOU WANT THE PASSWORD TO BE ECHOED ?"
      : INPUT "(Y/N) : ";ANS$
      : IF LEFT$(ANS$,1) <> "Y" THEN GOSUB 30010
      : GOTO 30580
30570 INPUT "ENTER YOUR PASSWORD : ";NAME$
30580 PRINT OPN$
      : PRINT WRT$;0
      : PRINT N
      : PRINT NAME$
      : PRINT AVAIL
30590 PRINT CLS$
      : PRINT "PASSWORD NOW CHANGED"
      : FOR I = 1 TO 1000
      : NEXT I

```

```

      : RETURN
30600 END
30610 PRINT OPN$
      : PRINT RDS$;I
      : INPUT X
      : IF X = 0 THEN PRINT "STUDENT HAS BEEN DELETED."
      : PRINT CLS$
      : RETURN
30620 INPUT NME$,LLEVEL,LDNE,LCORRECT,QLEVEL,QDNE,QCORRECT
30630 PRINT CLS$
      : PRINT "NAME           ";NME$
      : PRINT "LIN   LEVEL   ";LLEVEL
      : PRINT "      DONE      ";LDNE
      : PRINT "      CORRECT ";LCORRECT
      : PRINT "QUAD LEVEL   ";QLEVEL
      : PRINT "      DONE      ";QDNE
      : PRINT "      CORRECT ";QCORRECT
30640 PRINT
      : RETURN

```

APPENDIX B.

Program listing for teaching program.

Linear Equation Teaching Package.

```

80      ONERR GOTO 1000
90      PRINT CHR$(4); "BRUN LOMEM:"
      : & LOMEM: 16384
      : PRINT CHR$(4); "BLOAD HIGHER TEXT"
95      DIM E(5), T(5), TEMP(5)
      : DEF FN A(A) = INT (( RND (1) - 0.5) * A * 2)
100     D$ = CHR$(4)
      : FIL$ = " RE[ ]C[ ]E)OR[ ]U)R[ ]H)DS"
      : RDS$ = D$ + "READ" + FIL$ + ",R"
      : WRT$ = D$ + "WRITE" + FIL$ + ",R"
      : OPN$ = D$ + "OPEN" + FIL$ + ",L64"
      : CLS$ = D$ + "CLOSE" + FIL$
110     PRINT OPN$
      : PRINT RDS$;0
      : INPUT NAME$,NAME$,NAME$,NUM
      : PRINT RDS$;NUM
      : INPUT NAME$,NAME$,LEVEL,DNE,CRT,QL,QD,QC
      : PRINT CLS$
115     LS = LEVEL
      : DS = DNE
      : CS = CRT
120     CALL 3072
      : PRINT "(cL)"
      : V = 1
      : H = 1
      : C = 1
      : IF NAME$ = "" THEN N$ = ""
      : GOTO 170
130     C = C + 1
      : IF C > LEN (NAME$) THEN 160
140     IF MID$(NAME$,C,1) = " " THEN 160
150     GOTO 130
160     N$ = LEFT$(NAME$,1)
      : FOR I = 2 TO C - 1
      :   IF MID$(NAME$,I,1) >= "A" AND MID$(NAME$,I,1) <= "Z"
      :     THEN N$ = N$ + CHR$( ASC ( MID$(NAME$,I,1)) - ASC ("A")
      :       + ASC ("a"))
      : NEXT
170     PRINT "O.K. "N$","
      : PRINT "I will give you an example at the level you are at, then it will be your turn."
180     PRINT
      : PRINT "Type 'Q' if you don't want the example, any other key to continue : ";
190     POKE -16368,0
      : GET ANS$
      : IF ANS$ = "Q" THEN 220
210     GOSUB 1100
      : GOSUB 2000
      : REM EXAMPLE
220     PRINT "(cL)Now for a problem"
      : PRINT "Try this one ";N$

```



```

230  GOSUB 1100
    : GOSUB 1998
    : GOSUB 3000
235  IF LEVEL < 0 THEN LEVEL = 0
240  IF CRT = CS THEN 290
250  IF CRT = 1 THEN DNE = 1
255  IF LEVEL >= 100 THEN PRINT "(cL)You are at the      highest
    level !!!"
    : PRINT "well done "N$
    : LEVEL = 99
260  POKE - 16368,0
    : HTAB 1
    : VTAB 19
    : PRINT "(c0)Type 'Q' to stop,  any other letter formore prob
    lems : ";
    : GET ANS$
    : IF ANS$ = "Q" THEN PRINT "(cL)"
    : GOTO 10000
280  GOSUB 1100
    : GOSUB 3000
    : GOTO 240
290  PRINT "(cL)"
    : IF DNE = DS THEN PRINT "You got your first  problem wrong,"
    : GOTO 310
300  PRINT "You haven't got any right so far,"
310  VTAB 7
    : PRINT "Do you wish to go to a lower level ?"
    : PRINT "      ";
    : HTAB 1
    : INPUT "(Y / N):";ANS$
    : ANS$ = LEFT$(ANS$,1)
    : IF ANS$ = "N" THEN 280
320  IF ANS$ < > "Y" THEN 310
330  VTAB 13
    : PRINT "      ";
    : HTAB 1
    : INPUT "Enter new level:";ANS$
    : IF LEN(ANS$) > 2 THEN 330
340  X = VAL(ANS$)
    : IF X > LEVEL THEN 360
350  PRINT "(c0)The problems so far will not be counted"
    : PRINT "Good luck !"
    : DNE = DS
    : LEVEL = X
    : GOSUB 1999
    : GOTO 280
360  VTAB 15
    : PRINT "If you haven't got any correct at this level,how do yo
    u expect to get harder problems right ?";
    : HTAB 1
    : GOTO 330
1000 ZZ = PEEK(218) + PEEK(219) * 256
    : YY = PEEK(222)
    : IF ((ZZ < 3000) OR (ZZ > 4000 AND ZZ < 7000)) AND (YY = 255)

```

```

THEN PRINT "(cL)O.k. ";N$
      : PRINT "Here are some more problems."
      : GOTO 230
1010 PRINT "(cL)Error " PEEK (222)" in"
      : PRINT "statement " PEEK (218) + PEEK (219) * 256
      : GOTO 10005
1100 Y = LEVEL - INT (LEVEL / 10) * 10
      : X = FN A(2 * Y + 6)
      : IF ABS (X) <= 1 THEN 1100
1120 Y = LEVEL / 10 + 1
      : ON Y GOSUB 1150,1170,1200,1240,1250,1270,1280,1280,1290,1290
1130 RETURN
1140 REM LEVEL<10 ,Ax=B,NATURAL NUMBERS
1150 E(1) = ABS ( FN A(LEVEL * 2 + 4))
      : IF E(1) = 0 THEN 1150
      :
1155 X = ABS (X)
      : E(2) = X * E(1)
      : RETURN
1160 REM 10<=LEVEL<20,Ax=B,INTEGER NUMBERS
1170 E(1) = FN A(LEVEL / 2 + 2)
      : IF ABS (E(1)) <= 1 THEN 1170
1190 E(2) = E(1) * X
      : RETURN
1195 REM 20<=LEVEL<30, FRACTIONS
1200 E(3) = FN A(38) + 2
      : IF E(3) = X THEN 1200
1202 IF E(3) = 0 THEN 1200
1205 E(1) = E(3)
      : E(2) = X
      : EPS = 0.0001
      : MIN = ABS (E(1))
      : IF MIN > ABS (E(2)) THEN MIN = ABS (E(2))
1210 FOR I = 2 TO MIN
      : Y = E(1) / I
      : Z = E(2) / I
      : IF ( ABS (Y - INT (Y)) < EPS) AND ( ABS (Z - INT (Z)) < EPS
          ) THEN
      : E(1) = INT (Y + EPS)
      : E(2) = INT (Z + EPS)
      : MIN = INT (MIN / I + EPS)
1220 NEXT
      : IF E(2) < 0 THEN E(1) = - E(1)
      : E(2) = - E(2)
1230 RETURN
1235 REM 30<=LEVEL<40, DECIMAL NUMBERS
1240 E(1) = FN A(LEVEL - 26) * FN A(13) / 10
      : IF E(1) = 0 THEN 1240
1245 E(2) = E(1) * X
      : RETURN

```

```

1249 REM 40<=LEVEL<50, x+A=B, INTEGER
1250 E(1) = FN A(LEVEL - 20)
      : IF E(1) = 0 THEN 1250
1260 E(2) = E(1) + X
      : RETURN
1265 REM 50<=LEVEL<60, Ax+B=C, INTEGER
1270 E(1) = FN A(LEVEL - 46)
      : E(2) = FN A(20)
      : E(3) = E(1) * X + E(2)
      : IF E(1) = 0 OR E(2) = 0 OR E(1) = 1 THEN 1270
1275 RETURN
1279 REM 60<=LEVEL<80, Ax+B=Cx+D, INTEGER
1280 E(1) = FN A(LEVEL - 56)
      : E(3) = FN A(LEVEL - 56)
      : E(2) = FN A(25)
      : E(4) = (E(1) - E(3)) * X + E(2)
      : IF E(1) = E(3) OR E(1) = 0 OR E(2) = 0 OR E(3) = 0 OR E(4) =
        0 THEN 1280
1285 RETURN
1289 REM 80<=LEVEL<100, A(Bx+C)=D, INTEGER
1290 E(1) = FN A(24)
      : IF E(1) = 0 OR E(1) = 1 THEN 1290
1292 E(2) = FN A(20)
      : E(3) = FN A(30)
      : IF E(2) = 0 OR E(3) = 0 THEN 1292
1295 E(4) = E(1) * (E(2) * X + E(3))
      : RETURN
1998 FOR I = 1 TO 1000
      : NEXT I
      : RETURN
      : REM SHORT PAUSE
1999 FOR I = 1 TO 2000
      : NEXT I
      : RETURN
      : REM LONG PAUSE
2000 HCOLOR= 3
      : Y = LEVEL / 10 + 1
      : ON Y GOTO 2020,2080,2100,2130,2150,2220,2260,2260,2350,2350,6
        001
2010 VTAB 20
      : PRINT "Press any letter to continue : ";
      : POKE - 16368,0
      : GET ANS$
      : RETURN
2020 L = 14
      : ST$ = "To solve this"
2025 IF E(1) < > INT (E(1)) AND ABS (E(1)) < 1 THEN L = L - 14
2030 IF E(1) > 9.92 THEN L = L + 14
2035 IF E(1) < 0 THEN L = L + 14

```

```

2040 PRINT "(cL)"E(1)"x = "E(2);
    : H = PEEK (36)
    : PRINT
    : GOSUB 1999
    : VTAB 13
    : PRINT ST$;"",divide"
    : VTAB 15
    : PRINT "both sides by "E(1)
    : GOSUB 1998
2050 FOR I = 17 TO 19
    : HPLOT 0,I TO L,I
    : HPLOT L + 56,I TO H * 7,I
    : NEXT I
2060 VTAB 4
    : PRINT E(1)"      "E(1)
    : GOSUB 1999
2070 VTAB 17
    : PRINT "Giving..."
    : GOSUB 1998
    : VTAB 6
    : HTAB L / 7 + 1
    : PRINT "x = "E(2)"/"E(1)
    : GOSUB 1999
    : HTAB L / 7 + 1
    : PRINT " = "E(2) / E(1)
    : GOSUB 1999
    : GOTO 2010.
2080 GOTO 2020
2090 REM FRACTION EXAMPLE, 20<=LEVEL<30
2100 L = 14
    : PRINT "(cL)"E(1)"/"E(2)" x";
    : HTAB 16
    : PRINT "= "E(3)
    : GOSUB 1999
    :
2110 VTAB 13
    : PRINT "First multiply both sides by "E(2)
    : GOSUB 1998
    : VTAB 3
    : PRINT E(1)"/"E(2)" *"E(2)"x";
    : HTAB 16
    : PRINT "= "E(3)*"E(2)
    : GOSUB 1999
2120 VTAB 17
    : PRINT "Giving..."
    : GOSUB 1998
    : VTAB 5
    : E(2) = E(2) * E(3)
    : HTAB 8
    : PRINT E(1)"x";
    : HTAB 16
    : PRINT "= "E(2)
    : ST$ = "Now"
    : GOSUB 1999
    : GOSUB 2010
    : L = 14
    : GOTO 2030
2130 ST$ = "To solve this"
    : L = 42
    : IF E(1) = INT (E(1)) THEN L = 14

```

```

2140 GOTO 2025
2150 REM X+A=B.EXAMPLE
2160 SIGN$ = "+"
      : NUM$ = STR$ ( ABS (E(1)))
      : SN$ = "Take " + NUM$ + " from"
      : IF E(1) < 0 THEN SIGN$ = "-"
      : SN$ = "add " + NUM$ + " to"
2170 PRINT "(cL)x "SIGN$ " ABS (E(1))" = "E(2)
      : GOSUB 1999
2180 H = 12
      : IF ABS (E(1)) > 9 THEN H = 14
2190 VTAB 11
      : PRINT "First,"SN$
      : PRINT "both sides."
      : GOSUB 1998
      : VTAB 1
      : HTAB H
      : SIGN$ = CHR$ (88 - ASC (SIGN$))
      : PRINT SIGN$; ABS (E(1));
      : HTAB H + 16
      : PRINT SIGN$ " ABS (E(1))
      : GOSUB 1999
2200 VTAB 15
      : PRINT "Giving..."
      : GOSUB 1998
      : VTAB 3
      : PRINT "x + 0 = "E(2) - E(1)
      : GOSUB 1998
      : GOSUB 1998
      : VTAB 3
      : HTAB 5
      : PRINT " "
      : GOSUB 1999
      : GOTO 2010
2210 REM Ax+B=C EXAMPLE
2220 SIGN$ = "+"
      : NUM$ = STR$ ( ABS (E(2)))
      : SN$ = "take " + NUM$ + " from"
      :
      : IF E(2) < 0 THEN SIGN$ = "-"
      : SN$ = "add " + NUM$ + " to"
2230 PRINT "(cL)"E(1);
      : H = PEEK (36)
      : PRINT "x "SIGN$ " ABS (E(2));
      : L = PEEK (36) - H - 8
      : HTAB PEEK (36) + L + 2
      : PRINT " = "E(3);
      : K = PEEK (36)
      : GOSUB 1999
      : VTAB 11
      : HTAB 1
      : PRINT "First,"SN$
      : PRINT "both sides."
      : GOSUB 1998
2240 VTAB 1
      : HTAB L + H + 9
      : SIGN$ = CHR$ (88 - ASC (SIGN$))
      : PRINT SIGN$; ABS (E(2));
      : HTAB K + 1

```

```

: VTAB 1
: PRINT SIGN$; ABS (E(2))
: GOSUB 1999
: VTAB 15
: PRINT "Giving..."
: GOSUB 1998
2250 VTAB 3
: PRINT E(1)"x"; TAB( L + H + 6);" = "E(3) - E(2)
: GOSUB 1999
: GOSUB 2010
: ST$ = "Then"
: L = 14
: E(2) = E(3) - E(2)
: GOTO 2030
2260 REM Ax+B=Cx+D EXAMPLE
2270 S1$ = "+"
: IF E(2) < 0 THEN S1$ = "-"
2275 S2$ = "-"
: IF E(3) < 0 THEN S2$ = "+"
2280 S3$ = "+"
: IF E(4) < 0 THEN S3$ = "-"
2290 PRINT "(cL)"E(1)"x "S1$; ABS (E(2)); TAB( 16)" = "E(3)"x "S3$;
      ABS (E(4))
: GOSUB 1999
: VTAB 10
: PRINT "First put like terms together."
: GOSUB 1998
: S1$ = CHR$ (88 - ASC (S1$))
2300 VTAB 3
: PRINT E(1)"x"S2$; ABS (E(3))"x" TAB( 16)" = "E(4);S1$; ABS (E(
      2))
: GOSUB 1999
: VTAB 14
: PRINT "Giving..."
: GOSUB 1998
2310 VTAB 5
: PRINT "(("E(1);S2$; ABS (E(3))")x" TAB( 16)" = "E(4) - E(2)
:
: GOSUB 1999
: VTAB 16
: PRINT "Which gives"
: GOSUB 1998
2320 VTAB 7
: E(1) = E(1) - E(3)
: E(2) = E(4) - E(2)
: PRINT E(1)"x" TAB( 16)" = "E(2)
: L = 14
: ST$ = "Now"
: GOSUB 2010
: GOTO 2030
2340 REM A(Bx+C)=D EXAMPLE
2350 S$ = "+"
: IF E(3) < 0 THEN S$ = "-"
2360 PRINT "(cL)"E(1)"("E(2)"x "S$" " ABS (E(3))" ) = ";
: H = PEEK (36)
: PRINT E(4);
: K = PEEK (36)
: GOSUB 1999

```

```

      : VTAB 13
      : HTAB 1
      : PRINT "First divide both sides by "E(1)
      : GOSUB 1998
      : L = 14
      : IF ABS (E(1)) > 9 THEN L = 28
2370  IF E(1) < 0 THEN L = L + 14
2375  IF K < H THEN K = 40
2380  FOR I = 17 TO 19
      : HPLOT 0,I TO L,I
      : HPLOT H * 7,I TO K * 7,I
      : NEXT
2390  VTAB 4
      : PRINT E(1);
      : HTAB H + 1
      : PRINT E(1)
      : GOSUB 1999
      : VTAB 17
      : HTAB 1
      : PRINT "Giving..."
      : GOSUB 1998
      : HTAB L / 7 + 1
      : VTAB 6
      : PRINT E(2)"x "S$ " ABS (E(3))" = "E(4) / E(1)
2400  E(5) = E(1)
      : E(1) = E(2)
      : E(2) = E(3)
      : E(3) = E(4) / E(5)
      : GOSUB 2010
      : GOTO 2210
2900  PTCNT = 0
      : NUM$ = " "
2910  POKE - 16368,0
      : GET ANS$
      : IF ("0" <= ANS$) AND (ANS$ <= "9") OR (ANS$ = "-" AND NUM$
          = " ") THEN PRINT ANS$;
      : NUM$ = NUM$ + ANS$
      : GOTO 2910
2920  IF ANS$ = "H" THEN GOSUB 5000
      : POP
      : GOTO 3010
      : REM POP REMOVES CALL TO THIS ROUTINE AND THEN THE PROBLEM S
          TARTS ALL OVER AGAIN AFTER THE EXAMPLE HAS BEEN GIVEN.
2930  IF ANS$ = "." AND PTCNT = 0 THEN PRINT ANS$;
      : NUM$ = NUM$ + ANS$
      : PTCNT = 1
      : GOTO 2910
2940  IF (ANS$ = CHR$ (8)) AND ( LEN (NUM$) > 1) THEN HTAB PEEK (36)
          - 1
      : PRINT " ";
      : HTAB PEEK (36) - 1
      : IF RIGHT$ (NUM$,1) = "." THEN PTCNT = 0
2945  IF ANS$ = CHR$ (8) AND LEN (NUM$) > 1 THEN NUM$ = LEFT$ (NUM$,
          LEN (NUM$) - 1)
2950  IF ANS$ = CHR$ (13) THEN RETURN
2960  GOTO 2910

```

```

3000  FOR I = 1 TO 5
      :   TEMP(I) = E(I)
      : NEXT
      : WR = 0
      : DNE = DNE + 1
      : REM      PROBLEMS
3010  Y = LEVEL / 10 + 1
      : ON Y GOTO 3015,3015,3200,3015,3310,3390,3480,3480,3620,3620,6
          000
3015  PRINT "(cL)"E(1)"x = "E(2)
      : VTAB 19
      : PRINT "Type 'H' for help"
      : PRINT "Level="LEVEL" Correct="CRT
      : H = 2
      : IF E(1) > 9 THEN H = 4
3020  IF E(1) < > INT (E(1)) THEN H = H + 4
3025  IF E(1) < 0 THEN H = H + 2
3030  IF E(1) < > INT (E(1)) AND ABS (E(1)) < 1 THEN H = H - 2
3040  VTAB 4
      : HTAB H + 1
      : PRINT "x = ";E(2);"/";
      : L = PEEK (36) + 1
      : VTAB 10
      : HTAB 1
      : PRINT "Enter intermediate  answer"
      : VTAB 4
      : HTAB L
      : NUM$ = ""
      : PRINT "      ";
      : HTAB L
3050  GOSUB 2900
3060  IF VAL (NUM$) = E(1) THEN GOTO 3080
3070  GOSUB 6010
      : GOTO 3040
3080  VTAB 10
      : HTAB 1
      : PRINT "Enter final result"
      : PRINT "      "
      : VTAB 7
      : HTAB H + 5
      : PRINT "= ";
      : GOSUB 2900
3090  IF VAL (NUM$) = X THEN PRINT
      : HTAB 1
      : VTAB 10
      : PRINT "(c0)Well done "N$
      : CRT = CRT + 1
      : LEVEL = LEVEL + 2 - WR
      : GOSUB 1998
      : RETURN
3100  GOSUB 6010
      : GOTO 3080
3200  REM  20<=LEVEL<30, FRACTION PROBLEM
3210  PRINT "(cL)"E(1)"/"E(2)"x = "E(3)
      : VTAB 20
      : PRINT "Type 'H' for help"

```



```

      : PRINT "Level="LEVEL" Connect="CRT;
      : H = 2
      : IF ABS (E(1)) > 9 THEN H = 4
3220  IF E(1) < 0 THEN H = H + 2
3225  L = 2
      : IF E(2) > 9 THEN L = 4
3230  K = 2
      : IF ABS (E(3)) > 9 THEN K = 4
3235  IF E(3) < 0 THEN K = K + 2
3240  VTAB 10
      : HTAB 1
      : PRINT "Enter intermediate answer"
3250  VTAB 4
      : PRINT E(1)"/"E(2)"*" TAB( H + 2 * L + 6)"x = "E(3)"*"
      : VTAB 4
      : HTAB H + L + 5
3260  GOSUB 2900
      : VTAB 4
      : HTAB H + 2 * L + 6
      : N1$ = NUM$
      : PRINT "x ="E(3)"*";
      : GOSUB 2900
      : IF E(2) = VAL (N1$) AND E(2) = VAL (NUM$) THEN 3280
3270  GOSUB 6010
      : GOTO 3240
3280  VTAB 10
      : HTAB 1
      : PRINT "Which equals..."
      : HTAB H + 2 * L + 3
      : VTAB 7
      : PRINT "x = ";
      : HTAB 1
      : GOSUB 2900
      : HTAB H + 2 * L + 3
      : VTAB 7
      : PRINT "x = ";
      : N1$ = NUM$
3285  GOSUB 2900
      : HTAB 1
      : IF 1 = VAL (N1$) AND X = VAL (NUM$) THEN VTAB 10
      : HTAB 1
      : PRINT "(c0)Well done "N$
      : LEVEL = LEVEL + 3 - WR
      : CRT = CRT + 1
      : GOSUB 1998
      : RETURN
3290  IF E(1) = VAL (N1$) AND VAL (NUM$) = E(2) * E(3) THEN VTAB 10
      : HTAB 1
      : PRINT "O.K. "N$
      : GOSUB 1998
      : E(2) = E(2) * E(3)
      : GOTO 3015
3300  GOSUB 6010
      : GOTO 3280
3310  REM x+A=B PROBLEM
3320  SN$ = "+"

```

```

: S1$ = "-"
: IF E(1) < 0 THEN SN$ = "-"
: S1$ = "+"
3330 PRINT "(cL)x "SN$ " ABS (E(1))" = "E(2)
: VTAB 20
: PRINT "Type 'H' for help"
: PRINT "Level="LEVEL" Correct="CRT;
: H = 2
: IF ABS (E(1)) > 9 THEN H = 4
3340 HTAB 1
: VTAB 10
: PRINT "+ or - ?"
: VTAB 4
: PRINT "x "SN$ " ABS (E(1)); TAB( 10 + 2 * H)"= "E(2);
: HTAB H + 9
: GET ANS$
: PRINT ANS$;
: IF ANS$ = S1$ THEN 3360
3345 IF ANS$ = "H" THEN GOSUB 5000
: GOTO 3330
3350 GOSUB 6010
: GOTO 3330
3360 HTAB 1
: VTAB 10
: PRINT S1$ " what ? "
: VTAB 4
: HTAB H + 11
: GOSUB 2900
: VTAB 4
: HTAB 11 + 2 * H
: PRINT "= "E(2);S1$;
: N1$ = NUM$
: GOSUB 2900
: IF ( ABS (E(1)) < > VAL (N1$)) OR ( ABS (E(1)) < > VAL (NUM$)
) THEN GOSUB 6010
: GOTO 3360
3370 HTAB 1
: VTAB 10
: PRINT "Which gives "
: VTAB 7
: HTAB 2 * H + 7
: PRINT "x = ";
: GOSUB 2900
: IF VAL (NUM$) < > X THEN GOSUB 6010
: GOTO 3370
3380 HTAB 1
: VTAB 10
: PRINT "(cO)Well done "N$
: LEVEL = LEVEL + 2 - WR
: CRT = CRT + 1
: GOSUB 1998
: RETURN
3390 REM Ax+B=C PROBLEM
3400 SN$ = "+"
: S1$ = "-"
: IF E(2) < 0 THEN SN$ = "-"
: S1$ = "+"
3410 PRINT "(cL)"E(1)"x "SN$ " ABS (E(2))" = "E(3)
: VTAB 20
: PRINT "Type 'H' for help"

```

```

      : PRINT "Level="LEVEL" Correct="CRT;
      : H = 2
      : IF E(1) < 0 THEN H = 4
3420 K = 2
      : IF ABS (E(2)) > 9 THEN K = 4
3430 L = 2
      : IF E(3) < 0 THEN L = 4
3440 IF ABS (E(3)) > 9 THEN L = L + 2
3450 VTAB 10
      : HTAB 1
      : PRINT "+ or - ?"
      : VTAB 4
      : PRINT E(1)"x "SN$ " ABS (E(2)); TAB( H + 2 * K + 11)"= "E(3)
      ;
      : HTAB H + K + 9
      : GET ANS$
      : PRINT ANS$;
      : IF ANS$ = "H" THEN GOSUB 5000
      : GOTO 3430
3455 IF ANS$ < > S1$ THEN GOSUB 6010
      : GOTO 3450
3460 VTAB 10
      : HTAB 1
      : PRINT S1$ " what ? "
      : VTAB 4
      : HTAB H + K + 11
      : GOSUB 2900
      : VTAB 4
      : HTAB H + 2 * K + 13
      : N1$ = NUM$
      : PRINT "= "E(3);S1$;
      : GOSUB 2900
      : IF ( ABS (E(2)) < > VAL (N1$)) OR ( ABS (E(2)) < > VAL (NUM$)
          ) THEN GOSUB 6010
      : GOTO 3460
3465 VTAB 10
      : HTAB 1
      : PRINT "Giving ? "
      : VTAB 7
      : PRINT E(1)"x = ";
      : GOSUB 2900
      : IF E(3) - E(2) < > VAL (NUM$) THEN GOSUB 6010
      : GOTO 3465
3470 E(2) = E(3) - E(2)
      : HTAB 1
      : VTAB 10
      : PRINT "(c0)O.k. ";N$
      : GOSUB 1998
      : GOTO 3015
3480 REM Ax+B=Cx+D PROBLEM
3490 S1$ = "+"
      : S2$ = "-"
      : IF E(2) < 0 THEN S1$ = "-"
      : S2$ = "+"
3500 S3$ = "+"
      : S4$ = "-"
      : IF E(4) < 0 THEN S3$ = "-"
      : S4$ = "+"

```

```

3510 PRINT "(cL)"E(1)"x "S1$ " ABS (E(2))" = "E(3)"x "S3$ " ABS
      (E(4))
      : VTAB 20
      : PRINT "Type 'H' for help"
      : PRINT "Level="LEVEL" Correct="CRT;
      : H = 2
      : IF E(1) < 0 THEN H = 4
3520 IF ABS (E(1)) > 9 THEN H = H + 2
3525 K = 2
      : IF ABS (E(2)) > 9 THEN K = 4
3530 L = 2
      : IF ABS (E(3)) > 9 THEN L = 4
3540 IF E(3) < 0 THEN L = L + 2
3550 VTAB 10
      : HTAB 1
      : PRINT "+ or - ?"
      : VTAB 4
      : PRINT E(1)"x" TAB( K + H + 3)"x = "E(4);
      : HTAB H + 3
      : GET S$
      : PRINT S$;
      : IF S$ = "H" THEN GOSUB 5000
      : GOTO 3490
3555 IF (S$ < > "+") AND (S$ < > "-") THEN GOSUB 6010
      : GOTO 3550
3560 VTAB 10
      : HTAB 1
      : PRINT S$ " what ?"
      : VTAB 4
      : HTAB H + 5
      : GOSUB 2900
      : IF NOT ((S$ = "-" AND E(3) = VAL (NUM$)) OR (S$ = "+" AND E(3)
        ) = - VAL (NUM$)) THEN GOSUB 6010
      : GOTO 3550
3570 VTAB 10
      : HTAB 1
      : PRINT "+ or - ?"
      : VTAB 4
      : HTAB H + K + 5
      : PRINT "x = "E(4);
      : GET S$
      : PRINT S$;
      : L = PEEK (36)
      : IF S$ = "H" THEN GOSUB 5000
      : GOTO 3490
3575 IF S$ < > "-" AND S$ < > "+" THEN GOSUB 6010
      : GOTO 3570
3580 VTAB 10
      : HTAB 1
      : PRINT S$ " what ?"
      : VTAB 4
      : HTAB L + 1
      : GOSUB 2900
      : IF NOT ((S$ = "-" AND E(2) = VAL (NUM$)) OR (S$ = "+" AND E(2)
        ) = - VAL (NUM$)) THEN GOSUB 6010
      : GOTO 3570
3590 VTAB 10
      : HTAB 1
      : PRINT "Which gives ?"

```

```

: VTAB 7
: HTAB H + 7
: PRINT "x = ";
: HTAB H + 1
: GOSUB 2900
: N1$ = NUM$
: VTAB 7
: HTAB H + 7
: PRINT "x = ";
: GOSUB 2900
: IF N1$ = "1" AND X = VAL (NUM$) THEN VTAB 10
: HTAB 1
: PRINT "(c0)Well done "N$
: GOSUB 1998
: LEVEL = LEVEL + 2 - WR
: CRT = CRT + 1
: RETURN
3600 IF (E(1) - E(3) < > VAL (N1$)) OR (E(4) - E(2) < > VAL (NUM$))
      THEN GOSUB 6010
: GOTO 3590
3610 E(1) = E(1) - E(3)
: E(2) = E(4) - E(2)
: VTAB 10
: HTAB 1
: PRINT "(c0)O.k. "N$
: GOSUB 1998
: GOTO 3015
3620 REM A(Bx+C)=D PROBLEM
3630 S$ = "+"
: IF E(3) < 0 THEN S$ = "-"
3635 PRINT "(cL)"E(1)"("E(2)"x "S$" " ABS (E(3))")";
: H = PEEK (36)
: PRINT " = "E(4);
: L = PEEK (36)
: VTAB 20
: HTAB 1
: PRINT "Type 'H' for help"
: PRINT "Level="LEVEL" Correct="CRT;
: IF H > L THEN L = 40
3640 VTAB 10
: HTAB 1
: PRINT "Which operation      + - * /      ";
: HTAB 19
: GET ANS$
: PRINT ANS$
: IF ANS$ = "H" THEN GOSUB 5000
: GOTO 3635
3645 IF ANS$ < > "/" THEN GOSUB 6010
: GOTO 3640
3650 FOR I = 17 TO 19
:   HPLOT 0,I TO (H - 1) * 7,I
:   HPLOT (H + 4) * 7,I TO L * 7,I
: NEXT
: VTAB 10
: HTAB 1
: PRINT "Divide both sides      by what ?      "
: VTAB 4
: HTAB H / 2 - 2
: GOSUB 2900
: N1$ = NUM$
: VTAB 4
: HTAB H + 7

```

```

      : GOSUB 2900
      : IF E(1) < > VAL (NUM#) OR E(1) < > VAL (N1#) THEN GOSUB 6010
      : GOTO 3650
3660  VTAB 10
      : HTAB 1
      : PRINT "Which gives ?"
      : VTAB 7
      : PRINT E(2)"x "S# " ABS (E(3))" = ";
      : GOSUB 2900
      : IF E(4) / E(1) < > VAL (NUM#) THEN GOSUB 6010
      : GOTO 3660
3670  E(5) = E(1)
      : E(1) = E(2)
      : E(2) = E(3)
      : E(3) = E(4) / E(5)
      : VTAB 10
      : HTAB 1
      : PRINT "(c0)O.k. ";N#
      : GOSUB 1998
      : GOTO 3400
5000  FOR I = 1 TO 5
      :   T(I) = E(I)
      : NEXT
      : PRINT "(cL)Here is how its done"
      : GOSUB 1998
      : GOSUB 1100
      : GOSUB 2000
      : FOR I = 1 TO 7
      :   E(I) = T(I)
      : NEXT
      : RETURN
6000  PRINT "(cL)Sorry but there are no problems at this level."
      : PRINT "The level will be reset to 15 for you to do some pro
         blems."
      : LEVEL = 15
      : GOSUB 1999
      : GOTO 3010
6001  PRINT "(cL)Sorry but there are no examples at this level."
      : PRINT "There may be some problems to do."
      : GOSUB 1999
      : RETURN
6010  GOSUB 6030
      : VTAB 15
      : HTAB 1
      : PRINT "WRONG "N#
      : WR = WR + 1
      : IF WR = 3 THEN POP
      : PRINT "(c0)Here is how to do it"
      : GOSUB 1999
      : FOR I = 1 TO 5
      :   E(I) = TEMP(I)
      : NEXT
      : GOSUB 2000
      : LEVEL = LEVEL - 1
      : RETURN
6020  PRINT "Try again"
      : GOSUB 1998
      : VTAB 15
      : PRINT "
      : RETURN
6030  FOR I = 1 TO 50
      :   X = PEEK ( - 16336)

```

```


      :   FOR J = 1 TO 2
      :   NEXT J,I
      : RETURN
10000 POKE - 16368,0
      : INPUT "(cL)Are you sure you    wish to finish : ";ANS$
      : IF LEFT$(ANS$,1) < > "Y" THEN 280
10005 PRINT
      : PRINT OPN$
      : PRINT WRT$;NUM
      : PRINT "1"
      : PRINT NAME$
      : PRINT LEVEL
      : PRINT DNE
      : PRINT CRT
      : PRINT QL
      : PRINT QD
      : PRINT QC
      : PRINT CLS$
10010 CALL 3075
      : PRINT "O.K. "N$
      : PRINT "You did "DNE - DS" problems"
      : PRINT "You got "CRT - CS" right"
10020 IF DNE - DS < > 0 THEN PRINT "This is " INT ((CRT - CS) / (DNE
      - DS) * 1000 + 0.5) / 10"%"
      : IF LS > LEVEL THEN 10050
10030 IF LEVEL = LS THEN 10060
10040 PRINT "You went up by "LEVEL - LS
      : PRINT "levels to level "LEVEL
      : GOTO 10070
10050 PRINT "You didn't do well, going down "LS - LEVEL" levels to "
      LEVEL
      : GOTO 10070
10060 PRINT "You stayed at level "LEVEL
10070 END

```

APPENDIX C.

Sample output as shown by a dump of the
high resolution graphics screen.

$$-11(-6x + 9) = 165$$

Which operation
+ - * / 

Type 'H' for help
Level=95 Correct=0

$$\begin{array}{r} -11(-6x + 9) = 165 \\ \hline -11 \qquad \qquad -11 \\ -6x + 9 = -15 \\ \text{O.k. David} \end{array}$$

Fig (2).

Fig (1).
This shows the actual character size of Higher Text. Due to blurred edges on the pixels, the actual screen characters are more rounded than those above.

Fig (1) to Fig (7) show the sequence of steps required to solve a problem at level 95.

$$-6x + 9 = -15$$

$$-6x + 9 \blacksquare = -15$$

+ or - ?

Type 'H' for help
Level=95 Correct=0

Fig (3).

$$-6x + 9 = -15$$

$$-6x + 9 - 9 = -15 - 9 \blacksquare$$

- what ?

Fig (4).

Type 'H' for help
Level=95 Correct=0

$$-6x + 9 = -15$$

$$-6x + 9 - 9 = -15 - 9$$

$$-6x = -24 \blacksquare$$

Giving ?

Type 'H' for help
Level=95 Correct=0

Fig (5).

$$-6x = -24$$

$$x = -24 / -6$$

Enter intermediate
answer

Type 'H' for help
Level=95 Correct=0

Fig (6).

Fig (7).

$$-6x = -24$$

$$x = -24 / -6$$

$$= 4$$

Well done David

Type 'Q' to stop,
any other letter for
more problems :

Are you sure you
wish to finish : Y

O.K. David
You did 1 problems
You got 1 right
This is 100%
You went up by 2
levels to level 97

Fig (8).

As a safegaurd against a student
accidentally typing 'Q' when wanting
to do more problems, they are asked
if they are sure they wish to
finish, before giving their progress
for that session.

O.K. David,
I will give you an
example at the level
you are at, then it
will be your turn.

Type 'Q' if you
don't want the
example, any other
key to continue : ■

Fig (9).

This is the first display given
on the screen. From here, either
an example or a problem is given.

Fig (10).

The first screen of an example
at level 75, giving the order
in which lines come up to the
screen. Each line is displayed
with a pause between each
line.

1 $-13x - 24 = -15x - 8$
3 $-13x + 15x = -8 + 24$
5 $(-13 + 15)x = 16$
7 $2x = 16$
2 First put like terms
together.
4 Giving...
6 Which gives
8 Press any letter
to continue : ■

$$\begin{array}{rcl} \underline{2x} & = & \underline{16} \\ 2 & & 2 \\ x & = & 16/2 \\ & = & 8 \end{array}$$

Now, divide
both sides by 2
Giving...
Press any letter
to continue : ■

Fig (11).

The second screen of the above
example, also showing the order
in which lines were displayed
to the screen.

APPENDIX D.

Teacher documentation.

Teachers Section.

To gain access to this section.

To gain access to this section, boot the disk. You will first be asked if you have used this package before, to which you type 'Y'. You will then be asked for your student number, which is '0'. After this, you will be asked for your password, which is not echoed to the screen while you are typing it. The initial password when you get the disk is TEACHER, which you can change later if you wish.

Now in the teachers section.

After this, you are then in the teachers section section of the program, which allows you to access to the records to the records to see how each student is progressing. This is accomplished by a menu being displayed, and you typing in the number of the option you wish to do. The options are :

1). List progress for all students.

This first asks you whether you wish to have a hard copy printout of the output. Anything but a 'Y' in the first position is taken as a "no". If you type "yes" and there is no printer attached, the system will hang until you connect and turn on a printer, or press RESET. After the latter option, type 'RUN' and start again.

In either case above, the program will give a title, with column headings, and display up to eight student records per screen at two lines per student, pausing for

a character to be typed at the end of each screen. If 'Q' is typed at this point, the program will go back to the menu. When the last student has been displayed, you are asked to press RETURN when you are ready to return to the menu.

2). List progress for a range of students.

This is done the same as 1) above, except this option only displays progress for a range of student numbers. This first asks you the lower bound, and then the upper bound of the student numbers you wish to display. From here it goes to the section described in 1).

3). List progress for individual students.

For this, you type in the number of the student you wish to view the records of, followed by a RETURN. The progress of the student with that number will be displayed, if they haven't been deleted. Once you have finished listing the students you wish to see, type '0' to RETURN to the main menu.

4). Delete individual students.

When students have finished using the package and are not using it again, it is possible to delete each student, thereby making space available for other students. This can be done by typing the number of the student record you wish to delete. When you have finished deleting the students you wanted to, type '0' to return to the menu.

5). Delete all student records.

If you delete all student records by using 4), space will still be allocated on the disk for the maximum number of students who had been on the records since the last time this option was used. Using this option will actually delete the records file and reinitialize it with the record 0 of the file intact. Record 0 contains your password, the number of students using the package, the number of the last student to be deleted and the number of the student who last used the package. When this option is used, to prevent accidental erasure of the file, you are asked if you are sure you want to delete all the student's records. Anything but a 'Y' in the first position is taken as "no", and the records are not deleted.

6). Display number of students using the package.

This option takes the number recorded in record 0 as the number of students using the package and displays this at the terminal. If any students have been deleted, the number of deleted student records is not taken into account, as its primary use is to make sure students don't type in a number which is too big and beyond the end of the records file.

7). Change the password.

This option allows you to change the password to prevent students obtaining access to the records other

than that access which the program has to check and update student records when the student uses the package. It first asks if you want the password echoed to the terminal for verification, to which you type 'Y' or 'N', followed by a 'RETURN'. When you wish to use the package the next time, you will need to use this new password.

8). Exit to Applesoft

This just finishes execution of the program and returns you to Applesoft, with the program still in memory.

Recovery.

If for some reason, the records file becomes corrupted, or destroyed, it is not possible to gain access onto the system for either you or any of the students. In this case, to recover the file, type 'RUN LOAD', the file 'LOAD' being on the disk provided. This will delete the current record file, and create a new empty file with the password TEACHER. You will then be able to use the package again, although all students previously using the package will have to start again, as the original file is deleted.

APPENDIX E.

Student documentation.

Student Documentation.

What the package does.

This is a program which will try to teach you how to solve linear equations, by giving you first examples, then problems to work on and solve.

How to get on to the package.

The program will first ask you if you have used the package before. If it is the first time, you type 'N' followed by RETURN. The program will then ask you for your name. You now enter your first name followed by a space and then your surname. The program will then tell you your student number, which you will have to remember if you wish to use the package again and be able to start at the same place as you left off last time. Once you have remembered your number, type a letter to continue. You will then be asked which level you wish to start at. There are 100 levels, ranging from level 0 to level 99, 0 being the easiest, 99 the hardest.

The sort of problems you can expect at each level are :

Level	Problem	Types
=====	=====	=====
0 - 9	$ax = b$	all natural numbers.
10-19	$ax = b$	all integer.
20-29	$a/b \ x = c$	all integer.
30-39	$ax = b$	x integer. a,b real.
40-49	$x + a = b$	all integer.
50-59	$ax + b = c$	all integer.
60-79	$ax + b = cx + d$	all integer.
80-99	$a(bx + c) = d$	all integer.

If you have used the package before.

If you have used the package before, you will be asked for your student number. You will now enter the number which you were given the first time you used the package. After this you will be asked for your name to which you will type the same name you used when you first used this package.

Once you are on to the package.

Once you have done all the above which applies to you, you will then have to wait while another program is loaded. After it has been loaded, you will be asked if you wish to see an example of the type of question you will solve. If you have not tried any problems at this level, it would be advisable to have an example shown so you know how to solve this type of problem. If you do not want the example, type 'Q' and you will be given a problem straight away. If you do

ask for an example , the program pauses after it writes anything on the screen for you to read it, before showing the next line. If you want another example while doing a problem, type 'H' when asked for any input.

Some problems take more than one screen.

Some problems go through a series of steps to solve, each step taking a screen full to explain, so you will be asked to type a letter at the end of each screen full. Once the example has been given, you will then be given a problem to do.

What comes up to the screen.

For each problem, the equation is first displayed at the top of the screen, with the instruction "Type 'H' for help" and your progress as to which level you are at, and the total number of problems you have got correct so far. Diagrams showing what comes up to the screen can be found in Appendix C, Fig (1) to Fig (7). The first step in solving the problem will then be displayed and you will be asked for input. Three things can be typed in at this point, and any other time you are asked for input during a problem. They are :

1). 'H' for help.

This will then give you an example of the type of problem you are doing, returning to the start of the problem which you were doing when you typed 'H' once the example you are being given has finished.

2). An operator.

If you are asked for an operator, either +, -, *, or /, you will then type the operator which you think will be the one which can be used to solve the problem. The only other character which you can type at this point is 'H'. You do not press RETURN after this.

3). A number.

If it is none of the inputs above, then a number is expected. This number may be negative, in which case the - sign must be in the first position. If the number has a decimal point in it, there must be only one decimal point typed in. The program will not let anything but this input be entered. You can use the '<-' key to delete a number if you make a mistake, but only before you press RETURN.

Mistakes.

If you make a mistake, you will have another chance to enter the correct answer, except if you have already made two mistakes in attempting this problem already, in which case you will get the whole problem wrong and you will be shown how it should have been solved. If this happens, you will go down one level. If you get the problem correct, you will go up two levels minus the number of mistakes you made while doing the problem.

Object of each problem.

You will then step through the problem, until you find the value for 'x', possibly going through three screen fulls of working.

To stop getting more questions.

Once you have finished the problem, You will be asked to enter 'Q' to finish, any other letter to continue on with another problem. If you type anything other than 'Q', another problem will be given to you. If you type 'Q', the program will make sure you wish to stop, then it will display how well you did, and store your progress on disk for next time you use the program. To run the program again, turn the computer off, then back on again.

Stopping an example.

If you do not wish to see all of the example, you can type 'CTRL C' (to do this, hold down both the 'CTRL' and the 'C' key at the same time). This will give you another problem to do.