Computer Center

Writing Simple Review Programs

Richard Duhrkopf
Department Editor

I strongly believe that the best possible computer software is written "in house." That is, software we develop ourselves to satisfy our own specific needs. As I said in last month's column, we all have our own peculiarities in emphases and material. If we perceive that a topic will lend itself to computer assistance, then we are the best ones to design the software to explore that topic with our students. In most cases, that software is not written because of fear of programming rather than lack of ideas. So, I want to take this column to show how easy it is to program at least one type of software using BASIC.

First, we need to ask what topics are suitable for computer assistance. To me, there have always been three general types of educational software. The first type is review of classroom material, and that is the type of program that I want to discuss this month. The other two types are tutorials and simulations. Review programs usually present questions to the student. Those questions normally pertain to a specific classroom or text assignment, and they are aimed at helping the student determine to what level the material has been mastered. The student answers each question and is told whether the answer is correct or incorrect. In some cases, an incorrect response generates a reference to explanatory material or even explains the error to the student. This type of software usually emphasizes memory and recall, or what Piaget refers to as "concrete reasoning." Although some conceptual questions can be used, these programs work best with objective questions, especially multiple choice or true-false questions. However, questions that require an answer of a single word can also work well in these programs.

I like to design such programs around a single topic or lecture. I usually have ten questions in each program, and I try to make those questions build from the simple to the complex aspects of the topic. Students who have used these programs are enthusiastic about using them, and, in general, feel that they really help in testing their understanding of the material. I have found that a single program with 10 questions works well. That gives the students enough questions to make it worth their while, but does not cause them to sit for a long time at the computer. In addition, I like to liven the responses up a little. Others have criticized some of my programs because of that. However, I feel that I want to do everything that I can to make the students have fun using the computer. So, I might have them enter their name at the beginning and use their name occasionally in a response. Or, I might throw in some additional comments of encouragement.

Question and answer software has its good points and its bad points. I like it because it is easy to write. You do not need any fancy or elegant coding. It almost writes itself. You need a good understanding of the functioning of the print statements along with some understanding of branching. I do not mean to imply that such software is not important, or even fun to write; a lot of the software that I write is of this type. However, the programming is not difficult. If you are just beginning to program, I would recommend you start with programs like this. An easy way to do that is to take an old exam. For each question, use PRINT statements to generate the question on the screen, use an INPUT statement to get the answer from the student, and use an IF statement to evaluate the response and branch to the appropriate response to the answer. Then use more PRINT statements to respond to the student's answer. As a simple example, we can use the following BASIC segment for one question:

```
0010 PRINT "What is 3 + 2?"
0020 PRINT "A. 1"
0030 PRINT "B. 4"
0040 PRINT "C. 5"
0050 PRINT "D. 6"
0060 INPUT ANS$
0070 IF ANS$ = "C" THEN GOTO
     100
0080 PRINT "No. The correct answer
     is C. 3 + 2 = 5."
0090 GOTO 110
0100 PRINT "Yes. 3 + 3 = 5"
0110 . . .
```

Such software can be made more specific by branching. If you want to respond specifically to each wrong answer and identify the error made, that can be done easily. Taking our above example, we can see how this is done.

```
0010 PRINT "What is 3 + 2?"
0020 PRINT "A. 1"
0030 PRINT "B. 4"
0040 PRINT "C. 5"
0050 PRINT "D. 6"
0060 INPUT ANS$
0070 IF ANS$ = "B" THEN GOTO 86
0072 IF ANS$ = "C" THEN GOTO
     100
0074 IF ANS$ = "D" THEN GOTO
     104
0080 PRINT "No. The correct answer
     is C. 3 + 2 = 5."
0082 PRINT "The mistake that you
     made was that you subtracted 2
     from 3."
0084 GOTO 110
0086 PRINT "No. The correct answer
     is C. 3 + 2 = 5."
0088 PRINT "You made an error in
     your addition."
0090 GOTO 110
0100 PRINT "Yes. 2 + 3 = 5"
0102 GOTO 110
0104 PRINT "No. The correct answer
     is C. 3 + 2 = 5."
0106 PRINT "The mistake that you
     made was that you multiplied 2
     × 3."
0110 . . .
```

Notice that we never checked for

Richard Duhrkopf, editor of the Computer Center, is a lecturer in the **Department of Biology at Baylor University, Waco, TX 76798.** He teaches introductory biology for majors and nonmajors and serves as director of Biological Computing. He has a B.S. in Zoology and an M.S. and Ph.D. in Genetics, all from The Ohio State University. He has been active in the development of a wide variety of educational software.

the answer being A. That is taken care of by the code. Our three IF statements check to see if the answer was B, C or D and if it is, the program branches to the appropriate response. However, if the answer is not B, C, or D, the program executes the next statement which is the response for the A answer.

There are still several refinements that we can make to this. First, we have checked only for the input of upper case letters. We may want to allow the student to respond with either upper or lower case letters. In that case, an example of the proper coding would be

```
00** IF ANS$ = "A" or ANS$
   = "a" THEN GOTO . . .
```

In most versions of BASIC there is a function that can be used to convert all responses to upper case. In most, this function is called UCASE$, in others it is called UPCASE$. If your version of BASIC supports such a function, then the statement is

```
00** IF UCASE$(ANS$)
   = "A" THEN GOTO . . .
```

We can combine several if statements into a single logical statement by using ELSE. So, for our three if statements above, we can have one statement as follows:

```
0070  IF ANS$ = "B" THEN
GOTO 86 ELSE IF ANS$ = "C"
  THEN GOTO 100 ELSE IF
 ANS$ = "D" THEN GOTO 104
```

There are all kinds of other things that you can do. In one such group of programs that I was involved in writing with a group of people a few years ago, we had a wrong answer counter which kept track of successive wrong answers. We decided that if a student made three consecutive wrong answers, then there were problems. Either the student was not serious about doing the work, or the student was not prepared for the material. So, we counted successive wrong answers, and when they reached three, we had the program stop with a message to the student.

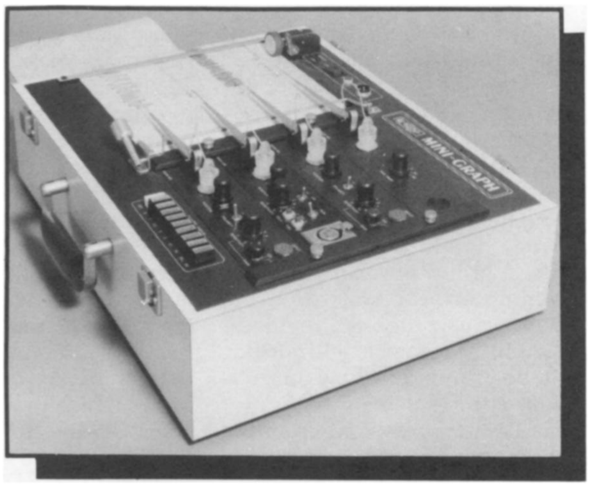
I also like to add a variable that counts the number of correct responses. At the beginning of the program, the value for that variable is set at 0, and when a correct response is made, the value of the variable is increased by 1. At the end of the program, the student is then told how many correct responses were made.

To me, of all of the different types of software, this type is the most idiosyncratic. It is like the study guides that I referred to in last month's column. For students in my classes, the question—answer—review software that I write is excellent because it helps them to identify the types of things that I feel are important. However, for students in other classes, the software is less important. Some of the aspects of a topic that I feel are important may not be as important to others, and they might feel that something I have left out is more important. However, I hope that I have demonstrated that these types of programs are really easy to write, and, if you want to begin to use the computer in this way, or want to get started in programming, these programs are very easy to produce for your students.

bring the lab into the classroom

With A Minigraph From Lafayette Instrument

Educators want a reliable, quality recording instrument for basic physiology experiments in the classroom that is easy to set-up and operate. Minigraphs from Lafayette Instrument are an economical choice for the academic biology lab. The Minigraph's modular design and wide selection of transducers and electrodes allow many human and animal physiological studies. All Minigraphs are backed by Lafayette Instrument's experience in manufacturing and servicing laboratory-quality data recorders.



76107-25 Deluxe Lab System priced at $3725.00
Complete systems starting under $1500.00

Basic Concepts & Experiments
in Physiology by Richard G. Planzer, Ph.D.

Lafayette Instrument Co.
P O Box 5729 Sagamore Parkway Lafayette, Indiana 47903 U.S.A.
Phone: 317-423-1505 TOLL FREE 800-428-7545 TELEX 276154 LICLAF

AD #0060