Computer Center

Organizing Review Programs

Richard Duhrkopf
Department Editor

In last month's column, I began a discussion of programming simple question and answer programs. As I described these types of programs last month, they are meant to provide a review for students. Normally, I do not like to present new material in these programs, just review material obtained either from class or the textbook. Last month, I went through the BASIC coding to write these programs using multiple choice questions. This month, I want to look at a few improvements and discuss some slightly different approaches and problems.

To begin, I want to talk about organizing these programs. The programming I presented last month is actually quite simple. The organization can be as simple or complex as you want to make it. I like to work in groups of ten questions. As such, I take a single, broad topic and decide which major principles I want to attack in the program. With 10 questions, I usually restrict that to three or four major principles. In that way, I have two or three questions for each principle. For example, in a group of questions covering simple Mendelian inheritance, I wanted to emphasize the principles of segregation, dominance and genotypic versus phenotypic ratios. That gave me three questions for each principle. For each of the principles, I could then ask a simple question, a moderate question and a more difficult question. In that way, the program could help to assess the level of understanding that the student has for each principle. I think it is better to write several different programs emphasizing smaller parts of a single topic than it is to write a single program that has many, broader questions. I know it may sound like a lot, but my goal is to have one program of 10 questions for each lecture. In a very real sense, these can replace the more traditional "Study Guides" sold to accompany many texts, and they are better for my students because they emphasize those things that I think are most important.

The next aspect is integrating the programs. Again, this can involve as much or as little programming as you want to do. To some extent, this also is influenced by what kind of computing is being done. On a large computer, using remote terminals, each group of 10 questions is a single program. At the beginning of the course, students are introduced to how to sign on to the system and how to get programs to run. For each lecture, the students are told (or it is written into the class schedule) to review by running a specific program.

On micros, I use a little more sophistication, but that is not really necessary. My approach is to group the programs together in broad topics on a single disk. The students could be told which program on what disk to run. However, I like to use the approach of "chaining" programs. The disk has the question and answer programs along with a common program which is used by the student to call and run the question and answer programs. The student begins by starting the calling (or "Driver") program. The Driver program presents the student with a menu from which the program to be run is chosen. For example, the BASIC code might be:

```
10 PRINT "Please choose one of the
following"
20 PRINT " 1. Bacterial Fission"
30 PRINT " 2. Mitosis"
40 PRINT " 3. Meiosis"
50 PRINT " 4. Quit"
60 INPUT ANS%
70 ON ANS% GOTO 80,90,100,110
80 LOAD "BACQUEST", R
90 LOAD "MITQUEST", R
100 LOAD "MEIQUEST", R
110 END
```

The above BASIC coding contains two new types of statements that, if you are new to BASIC programming, we need to discuss briefly. The first is the ON . . . GOTO . . . statement. This is a way of evaluating and branching based upon the value of a variable. In this case, the variable is ANS% (the % is a symbol used in BASIC variables to indicate that the value of the variable will be an integer). This statement is equivalent to four IF . . . GOTO statements. It says IF ANS% = 1 THEN GOTO 80, IF ANS% = 2 THEN GOTO 90, IF ANS% = 3 THEN GOTO 100, and IF ANS% = 4 THEN GOTO 110. In an ON . . . GOTO . . . statement, you should have as many branching references as there are possible values of the variable. In this case, there are four possible values that the student can enter, and four possible branches.

The second new statement is the LOAD ". . .", R statement. This statement varies with different dialects of BASIC. So, please consult your reference manual for the proper form. The form that I have used is for MBASIC. The statement tells the computer to load the file whose name is found in quotes into memory. The R is short for RUN. So, you are telling the computer to load the program whose name is in quotes into memory and run it. As I mentioned above, this statement can take a variety of forms. The ZBASIC form of the statement is OPEN "I",1,". . .": RUN 1. This is actually a more conventional way of telling the computer what to do. A third syntax that I have seen is just RUN ". . .". All three of these forms accomplish the same goal of calling and running one program from another.

By chaining programs like this, the student does not need to remember the specific name of the program that they are supposed to run, only the topic. In addition, with more programming, you can keep better track of the student's progress. You can inform a student if they have already

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.

run that program, and ask them if they really want to run it again or a variety of other things. (That will be a topic for a future column.)

Finally, at the end of the question and answer program, you want to return them to the Driver program using a similar LOAD ". . .", R statement. In that way, the student always enters and exits via a single program.

Another, more complicated approach to these types of question and answer programs is to use the program to evaluate the student's understanding of a topic, and take appropriate steps. You may want to write a larger program with more questions and decide in the program whether or not some questions are to be asked. You could begin with the standard 10 questions. However, for each question, you might have four additional questions on the same topic. If students get the first question correct, they are branched to the first question in the next topic. If they do not get the first question correct, they are branched to the next question covering that same topic. The response to the second question then determines if they get the third question or are branched to the first question of the next topic. Thus, students may get anywhere from 10 to 50 questions, depending upon how many questions they answer correctly. The actual programming for this is no more complex than we saw last month; it only involves writing larger programs with a little more branching.

Finally, our programs so far have involved only objective, multiple choice questions. It should be obvious that true-false questions will work just as well. Short answer questions are a little more difficult. If you are expecting the student to enter a specific word from the keyboard, there are a couple of things to keep in mind. By using the UCASE statement we discussed last month, you can eliminate problems of capitalization. However, you must determine to what extent the response can be misspelled. Normally, you may want to allow for one or two common errors in the spelling of the word that you are looking for. Otherwise, it can be quite frustrating for the student who enters what is thought to be the correct word, only to find that the word was misspelled and considered by the computer to be wrong. I have found this to be serious enough that I do not use such questions. If I do want to ask such a question, I can usually force it into a multiple choice format to avoid such problems.

AV Reviews

Rachel Hays
Department Editor

The earthworm: Darwin's plow. (Science-Biology/Life Series). 1985. Coronet Film and Video, Northbrook, IL. 16mm color-sound film. 12.5 min. Purchase $350.

This is a rather short, interesting, nontraditional presentation of the earthworm. Instead of just discussing the various earthworm systems, it uses Charles Darwin's research on the annelid as the theme around which to orient the presentation. This alone is an important feature of the film, since most students are only aware of Darwin's work on natural selection. It even shows a problem that Darwin couldn't solve: how the earthworm detects light. The photography showing earthworm movement is excellent. However, it is puzzling that the film did not explain the reason for using red light for these photographic sequences.

The inquiry method is demonstrated following a typical dissection of the earthworm's gizzard by looking at the gizzard contents under a microscope. Since leaf particles are found, time-lapse photography was used to examine the earthworm's diet by providing a leaf as food. After showing that earthworms eat and excrete humus, thereby recycling nutrients, an estimate is made that today there are one million earthworms per farming acre recycling approximately 100 tons per pear. Earthworm burrowing behavior is graphically shown when hot wax is poured into empty earthworm tunnels, and then excavated when cool. Earthworm reproductive behavior is demonstrated with time-lapse photography.

A complete teacher's guide is included, consisting of a summary, background information, learning objectives and questions for before and after showing the film. This film would be most appropriate for a junior high audience, although it could be used with high school students as well.

Edward B. Ruth
Milton Hershey School
Hershey, PA 17033-0830

The Portugese man-of-war. 1986. Carolina Biological Supply Co., Burlington, NC. Video. 9 min. Purchase: $119.95.

This color video depicts the characteristics of the Portugese man-of-war and certain of its ecological relationships. The video emphasizes the sting cells and other cnidarian characteristics. It also shows and contrasts the polyp and medusae forms of life within the colony and describes the colony's feeding behavior, reproductive methods, enemies and habitat.

The lack of specialized vocabulary and the stress on natural history of the colony make this suitable for use with students of all ages. A brief teacher's guide suggests uses for the video and provides questions and references. Teachers who emphasize invertebrate zoology and/or ecological relationships will find this video useful.

Gerald Skoog
Texas Tech University
Lubbock, TX 79413

Forests across the United States. 1984. Library Filmstrip Center, Bloomington, IL. Sound-filmstrip. 14 min. Purchase: $35.

This filmstrip is a survey of several different forest communities in America. It is a well illustrated program with 76 color frames. These are uniformly high quality photographs that take us to rain forests in the Pacific Northwest, coniferous forests in the Midwest, cypress swamps in Louisiana, and through the Appalachians to Maine. Locator maps precede each forest discussed, helping to separate segments of the filmstrip. Most of the photographs are of moun-

Rachel Hays is the editor of the Audio Visual Review section of ABT. She teaches science at Heath Junior High School, in Colorado's Weld County School District #6. She holds a Ph.D. in Botany from the University of California, Davis, and has taught courses at the college level. With a B.S. from San Diego State University, Hays went on to the University of California, Davis for her M.S. degree. For several years, Hays has done research for the Natural Resources Ecology Laboratory at Fort Collins, CO, studying nutrient cycling and soil organisms. She has published articles in several popular and scientific periodicals. Her address is: **6921 Buckhorn Ct., Loveland, CO 80537.**