## Cleveland State University EngagedScholarship@CSU



**Physics Faculty Publications** 

Physics Department

12-1-1987

## Generating Problem Sets with Word Processing Software

Jearl D. Walker *Cleveland State University* 

Follow this and additional works at: https://engagedscholarship.csuohio.edu/sciphysics\_facpub
Part of the <u>Physics Commons</u>
How does access to this work benefit you? Let us know!
Publisher's Statement
Copyright 1987 American Association of Physics Teachers. The article appeared in American Journal of Physics 55 (1987): 1141-1142 and may be found at http://aapt.scitation.org/doi/10.1119/

1.15264

## **Original Citation**

Walker, Jearl D. "Generating Problem Sets with Word Processing Software." American Journal of Physics 55 (1987): 1141-1142.

## **Repository Citation**

Walker, Jearl D., "Generating Problem Sets with Word Processing Software" (1987). *Physics Faculty Publications*. 133. https://engagedscholarship.csuohio.edu/sciphysics\_facpub/133

This Article is brought to you for free and open access by the Physics Department at EngagedScholarship@CSU. It has been accepted for inclusion in Physics Faculty Publications by an authorized administrator of EngagedScholarship@CSU. For more information, please contact library.es@csuohio.edu.

Jearl Walker

Problem sets generated by a computer have been described previously in this Journal. 1-3 They consist of multiple-choice questions where the data within the problems and the arrays of one or more correct answers and several wrong answers are changed every time the problem sets are run with a new "random seed." The advantage of the problem sets is that each new class of students must work the problems to determine the correct answers and cannot merely record the correct answers from a previous class. The disadvantage of the problem sets is that they must be written within a program language such as BASIC because printing instructions must be included. This procedure makes editing of the text cumbersome. If, for example, I find that a problem could be clearer with additional instructions, I must rewrite the program controlling that section of the problem set.

I now employ a simpler procedure through the use of a word processing software that has a "mail merge" capability. It allows the full editing capability provided by a word processor and requires no programming. The software Xywrite III<sup>4</sup> runs on the IBM PC and any of its "clones." I write the text of a problem set as I would any other material except that where data is to be inserted I introduce a special marker with a number. These markers are called "put fields" and the number tells the computer the location of the data in a separate list.

I prepare a separate data list for each different version of the problem sets I require. I have ten versions. After ten sets of students have come through my course, I begin the cycle again.

A template for a problem set looks like Fig. 1. (The actual screen appearance of the text has the PF# in a "bold reverse" format and a prefix triangle that is not reproduced here.) The data list looks like Fig. 2. (The leftward going arrow is Xywrite's symbol for the end of a line.)

For brevity, only the first 14 data items are shown in Fig. 2. Notice that the data can be inserted at more than one location in a problem set. The answer array in the problem set template looks misaligned but since each answer in the data list has ten spaces (whether filled or not), the printed

1.01 A car with a speed of PF1 m/s travels for PF2 s. How far has it moved?

(a) PF3 (b) PF4 (c) PF5 (d) PF6 (e) PF7

(f) PF8 (g) PF9 (h) PF10 (i) PF11 (j) PF12 m

1.02 If a car has the same speed of PF1 m/s, but travels for PF14 s, how far has it moved?

(a) PF15 (b) PF16 (c) PF17 (d) PF18 (e) PF19

(f) PF20 (g) PF21 (h) PF21 (i) PF22 (j) PF23 m

copy of the answer array is neatly aligned. The data list can have any numerical or word data. For example, I can insert "rightward" into a problem one year and "leftward" the next year.

The 13th item in the data list happens to be the answer to the first question. It includes the problem number, correct choice of an answer, and any message I want. I might add notes as reminders to emphasize certain aspects in the problem when I work it for the students. The template for an answer key looks like Fig. 3. When the file is printed, only the answers and messages are listed. This listing is handy because my course is taught in a lecture-recitation mode. At the beginning of the course I distribute a problem set booklet and the printed answer keys for that year to each of the recitation teachers.

Once the files for the problems and answer keys are prepared, they need be changed only if I want to fine tune the problem sets or add extra messages. The data files differ not only in the numerical and word data but also in the location of the correct answer in each answer array.

To print a copy of the problem sets I command Xywrite to merge a particular data file into the problem set template and print the result. Similarly I command it to merge the data file into the answer key template and print the result. This merge procedure is the function of mail merge.

The result of merging can be seen on the screen prior to printing by commanding Xywrite to "review" or "type to screen" the material. With a review I can tidy up the appearance of the problem sets by noting where the pages end and then forcing my own page-end commands into the template. For example, I dislike having only one of the two lines of possible answers printed at the bottom of a page, with the second line at the top of the next page. When I discover such a split in the review, I force a page-end command into the template for the problem set immediately before the first line of answers or before the problem itself. Unless I change the problem sets later, this preparation need be done only once.

Xywrite and some other software packages allow several files to be seen simultaneously on the screen. When I type a

12 ←	
4-	
36	÷-
24	←
30	+
38	÷-
48	←
22	<b>←</b>
40	⊷
42	⊷
44	+
46	+
1.01 E	$d = vt \leftarrow$
6⊷	

Fig. 2. Data listing

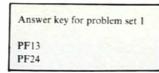


Fig. 3. Answer key template.

data list, I also have the template for the problem set on the screen so that I can determine the sequencing of data. If I later decide to introduce a new problem within a problem set, I do not need to renumber all of the put-fields. Instead I add the appropriate data at the end of a data list and then embed within the new problem the put-fields with numbers that instruct the computer to look at the end of the list for the data.

Instead of manually typing the data list, I could arrange for it to be generated by a program working with a random seed as has been previously described. This procedure would eliminate the need for typing out multiple versions of the data list. By running the program ten times, I would have ten versions.

I have employed multiple-choice problem sets generated and graded by a computer for 12 years. In each of three academic quarters of the year-long course of introductory physics the students must answer about 400 problems. I begin the year with about 200 students. Since I invest time only once in writing the problems, the ratio of the students' work to my work is quite high. Professors teaching the evening, summer, and "trailer" offerings of the course also use the problem sets because the sets demand a great deal of work by the students and little work by the professors.

<sup>1</sup>J. Walker, V. F. Walters, and K. J. Casper, Am. J. Phys. 45, 518 (1977).

<sup>2</sup>G. T. Wood and K. J. Casper, Am. J. Phys. 47, 388 (1979).

<sup>3</sup>G. K. Miner and J. E. Lang, Am. J. Phys. 52, 316 (1984).

<sup>4</sup>XyQuest, Inc., P.O. Box 372, Bedford, MA 01730. Xywrite III is a powerful word processing software that includes mail merge, custom-designed commands, split-screen editing, multiple-screen files, and batch printing. It can also search through an entire disk for a given symbol or phrase. Since it avoids menus, it is extremely fast in execution.