1990-06

Guide to develop a refresher for MA1117, single variable calculus

Lampugnano, Matthew

Monterey, California: Naval Postgraduate School

http://hdl.handle.net/10945/27776

2

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

Guide to Develop a Refresher for MA1117,
Single Variable Calculus

by

Matthew Lampugnano

June 1990

Thesis Advisor: Gordon E. Latta

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION
Unclassified

1b. RESTRICTIVE MARKINGS

2a. SECURITY CLASSIFICATION AUTHORITY

2b. DECLASSIFICATION / DOWNGRADING SCHEDULE

3. DISTRIBUTION / AVAILABILITY OF REPORT
Approved for public release;
distribution is unlimited.

4. PERFORMING ORGANIZATION REPORT NUMBER(S)

5. MONITORING ORGANIZATION REPORT NUMBER(S)

6a. NAME OF PERFORMING ORGANIZATION
Naval Postgraduate
School

6b. OFFICE SYMBOL (*If applicable*)
380

7a. NAME OF MONITORING ORGANIZATION
Naval Postgraduate School

6c. ADDRESS (*City, State, and ZIP Code*)
Monterey, CA. 93943-5000

7b. ADDRESS (*City, State, and ZIP Code*)
Monterey, CA. 93943-5000

8a. NAME OF FUNDING / SPONSORING ORGANIZATION

8b. OFFICE SYMBOL (*If applicable*)

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER

8c. ADDRESS (*City, State, and ZIP Code*)

10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.

11. TITLE (*Include Security Classification*)
GUIDE TO DEVELOP A REFRESHER FOR MA1117, SINGLE VARIABLE CALCULUS

12. PERSONAL AUTHOR(S)
Lampugnano, Matthew

13a. TYPE OF REPORT
Master's Thesis

13b. TIME COVERED
FROM ________ TO ________

14. DATE OF REPORT (*Year, Month, Day*)
June 1990

15. PAGE COUNT
146

16. SUPPLEMENTARY NOTATION The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U. S. Government.

17. COSATI CODES

FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (*Continue on reverse if necessary and identify by block number*)
Refresher for MA1117, Single Variable Calculus

19. ABSTRACT (*Continue on reverse if necessary and identify by block number*)

Refreshers for introductory courses have a variety of useful purposes. They may be used as a tool for newly arriving students to assist in their return to an academic environment, as a review for tests, or as a prelude to what a course offers. They may also be sent to interested personnel in the field. The primary benefit of the refresher is to experience faster learning and greater retention of the material covered.

This thesis is a step by step instruction of how to develop a microcomputer based refresher for any subject. These refreshers, in the form of a series of questions and answers, are easy to develop as well as easy to use. A Zenith-248 microcomputer or compatible is the main tool used to develop the refresher. An initial file, written on a word processor containing the questions and answers, is the raw data. By following a few simple instructions when creating this file, it can be transformed into a refresher in a minimal amount of time. A refresher for MA1117, single variable calculus, is developed as an example.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION
Unclassified

22a. NAME OF RESPONSIBLE INDIVIDUAL
Gordon E. Latta

22b. TELEPHONE (*Include Area Code*)
408-646-2206

22c. OFFICE SYMBOL
LZ

83 APR edition may be used until exhausted
All other editions are obsolete

☆ U.S. Government Printing Office: 1986—606-24.

Guide to Develop a Refresher for MA1117, Single Variable Calculus

by

Matthew Lampugnano
Captain, United States Marine Corps
B. S., United States Naval Academy, 1980

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN APPLIED MATHEMATICS

from the

NAVAL POSTGRADUATE SCHOOL

June, 1990

Author: [redacted]
Matthew Lampugnano

Approved by: [redacted]
Gordon E. Latta, Thesis Advisor

[redacted]
Donald A. Danielson, Second Reader

[redacted]
Harold M. Fredricksen, Chairman
Department of Mathematics

ABSTRACT

Refreshers for introductory courses have a variety of useful purposes. They may be used as a tool for newly arriving students to assist in their return to an academic environment, as a review for tests, or as a prelude to what a course offers. They may also be sent to interested personnel in the field. The primary benefit of the refresher is to experience faster learning and greater retention of the material covered.

This thesis is a step by step instruction of how to develop a microcomputer based refresher for any subject. These refreshers, in the form of a series of questions and answers, are easy to develop as well as easy to use. A Zenith-248 microcomputer or compatible is the main tool used to develop the refresher. An initial file, written on a word processor containing the questions and answers, is the raw data. By following a few simple instructions when creating this file, it can be transformed into a refresher in a minimal amount of time. A refresher for MA1117, single variable calculus, is developed as an example.

TABLE OF CONTENTS

ACKNOWLEDGMENT

I wish to thank my thesis advisor, Professor Gordon Latta for his expert opinion and advice. All assembly language programs were written by Professor Latta. Only through his dedicated efforts and constant encouragement was this thesis completed.

I. INTRODUCTION

The transition from military officer to graduate student is one that many newly arriving Naval Postgraduate School students find most challenging. As many students have been out of an academic environment for a number of years, they may rely on refreshers to assist in making the transition as smooth as possible. Simply put, a refresher is a series of questions and answers which highlight the objectives of the course. For ease of use, the refresher is developed and run on a Zenith-248 microcomputer or compatible machine.

The purpose of this thesis is twofold. First, a quick and easy procedure is developed so that anyone may produce a refresher. The second part of the thesis is the actual refresher for single variable calculus, MA1117. The questions and answers for this refresher are based on the book *Calculus*, Second Edition, by Dennis D. Berkey. The refresher will then be administered to students enrolled in the MA1117 course to help them in their mastering of the material.

A refresher has many useful purposes. A refresher may be used to determine the level of a student's knowledge in order that he be properly placed in the introductory courses (level 1000 and 2000). Using the refresher, the student may determine whether or not he feels comfortable with the material. If the student cannot answer the questions confidently, then he should take the course. If the student believes that the material has been mastered, then he should validate the course and move on to the next level.

Another important use of the refresher is its value as a tool for reviewing the material in the course. This can be accomplished in one of two ways. First, since the questions in the refresher are based on the course objectives, the refresher can be used as an excellent review for a midterm or final exam. Second, once the course has been completed, the refresher can be used to review concepts previously mastered.

Lastly, the refresher can be used by students to decide whether or not to take the course, mainly as an elective. Since the major topics of the course are covered, students can look at the refresher to see if they are interested in the subject matter. This should facilitate the students' decision to enroll in the course.

II. NATURE OF PROBLEM

The refresher, when running on a Zenith-248 or compatible microcomputer, is a series of questions and proposed answers. Each question and proposed answers appear in random order on the screen in yellow starting in the upper left corner. The student reads the question and tries to decide the correct answer. By pressing any key, the correct answer along with any pertinent information appears in red brackets under the proposed answers. By pressing any second key, the next question appears on the screen and the process continues. The process is ended by pressing the control C(^C) at any time. By cycling through the questions in random order, the student should obtain a feel for the material. Graphs or pictures may also be incorporated in the questions to assist in learning the subject material.

The fundamental problem is to develop a framework in which anyone can write the refreshers to be placed on disks for use on a Zenith-248 or compatible microcomputer. One begins with the raw data of questions and proposed answers, along with the correct answers. Then a series of programs, written in assembly language, input the raw data into the refresher. The raw data is entered into a document file using a word processor. By following a simple set of instructions, the author of the questions and answers can proceed quickly to the refresher. Within this framework, a refresher for MA1117, single variable calculus, is developed as an example.

III. PROCEDURE

The following items are essential to develop the refresher:

- A Zenith-248 microcomputer or a compatible machine
- A word processor, used to enter the question and answers and to make any changes to the assembly language programs that were written to develop the refresher (e.g., *WORDSTAR*)
- A copy of the following machine language programs:
  - `strip.com`
  - `contc.com`
  - `align.exe`
  - `count.com`
  - `hexasc.com`
  - `resident.com`
  - `display.com`
  - `testtemp.asm`
  - `template.asm`
  - `masm.exe`
- If graphs are desired, then the graphs should be of the format of a highlighted background with the actual graph or pictures in black. *PCPAINT* with a mouse is ideal for this purpose. Also, simple basic programs can be written to

perform the same objective. Graphs are displayed in mode 4 (320 pixels by 200 pixels).

The following nine steps are used to develop the refresher.

STEP 1: Writing the questions and answers on a wordprocessor.

This step is the hardest and most time consuming of all the steps in the process and should be taken with great care. The questions in the Calculus refresher are designed to be answered with minimal computations, if any. Definitions, theorems, and simple computational problems are the main focus of the Calculus refresher. The refresher uses two control characters to control the flow of the program. The square brackets, [and], are used to bracket the correct answers and * is used to determine where graphs are to be located. (Sometimes * is not a convenient symbol to use for graph control since it may be used in the questions and answers, as a symbol for multiplication for example. If this is the case, then another character that will not be used in any question or answers such as \ or @ can be used for the graph control character in place of *. This new character is used to identify which questions have a graph associated. In Step 8 a slight change is made so that the refresher recognizes this new control character). Therefore, when writing questions and proposed answers on the word processor, the characters [,], and * must not be avoided.

When typing the questions and proposed answers using the word processor, the correct answer should be placed between the brackets, [and], three lines after the last proposed answer. Any amount of information can be placed between the brackets, including a reference for the answer (see Figure 3.1).

Graphs may also be displayed with the questions to assist in developing an idea. These are easily incorporated into the refresher by placing the control character * for graph control in the space preceding the left bracket of the answer. If more than one graph is needed for a question, then place the same number of graph control characters (*'s) before the left bracket of the correct answer (see Figure 3.2).

The right bracket of the correct answer is the control character for where that question ends and the following question begins. When using the word processor to input questions and answers, if the next question begins on the line immediately after the line containing the correct answer to the previous question, then the questions will appear on the screen starting in the upper left corner. If a different starting point is desired, say five lines from the top of the screen, then one should leave five blank lines between the previous correct answer and the start of the new question. These blank lines have the effect of moving the question down the screen. (see Figures 3.3 and 3.4)

When writing the questions and answers, sometimes it is easier to work on several shorter files. Each file, or submodule, contains a specific topic such as integration, differentiation, power series, etc. These submodules can then be concatenated into one larger file. This approach is very reasonable and also very flexible. When each submodule is completed, the following DOS command is given to concatenate each submodule into the final file containing all the questions and answers:

```
COPY FILE1.DOC+FILE2.DOC+FILE3.DOC+FILE4.DOC FINAL.DOC
```

FINAL.DOC is the file which the assembly language programs will manipulate to produce the refresher. As one final check, use the word processor to make sure that where the files were joined together, the questions run in sequential order and that the spacing between questions is correct.

In summary, the output of this step is a file called FINAL.DOC. Contained in this file are all questions and possible answers followed in square brackets by the correct answers and any other pertinent information.

1. Which of the following describes an even function?

a) $f(x) = f(-x)$

b) $f(x) = -f(-x)$

c) $f(x) = -f(x)$

d) none of the above

[a] or [a, ref. page 123, Berkey]

(Either answer in the square brackets is correct.)

Figure 3.1

2. Which of the following graphs is an even function?

a.) A

b.) B

c.) C

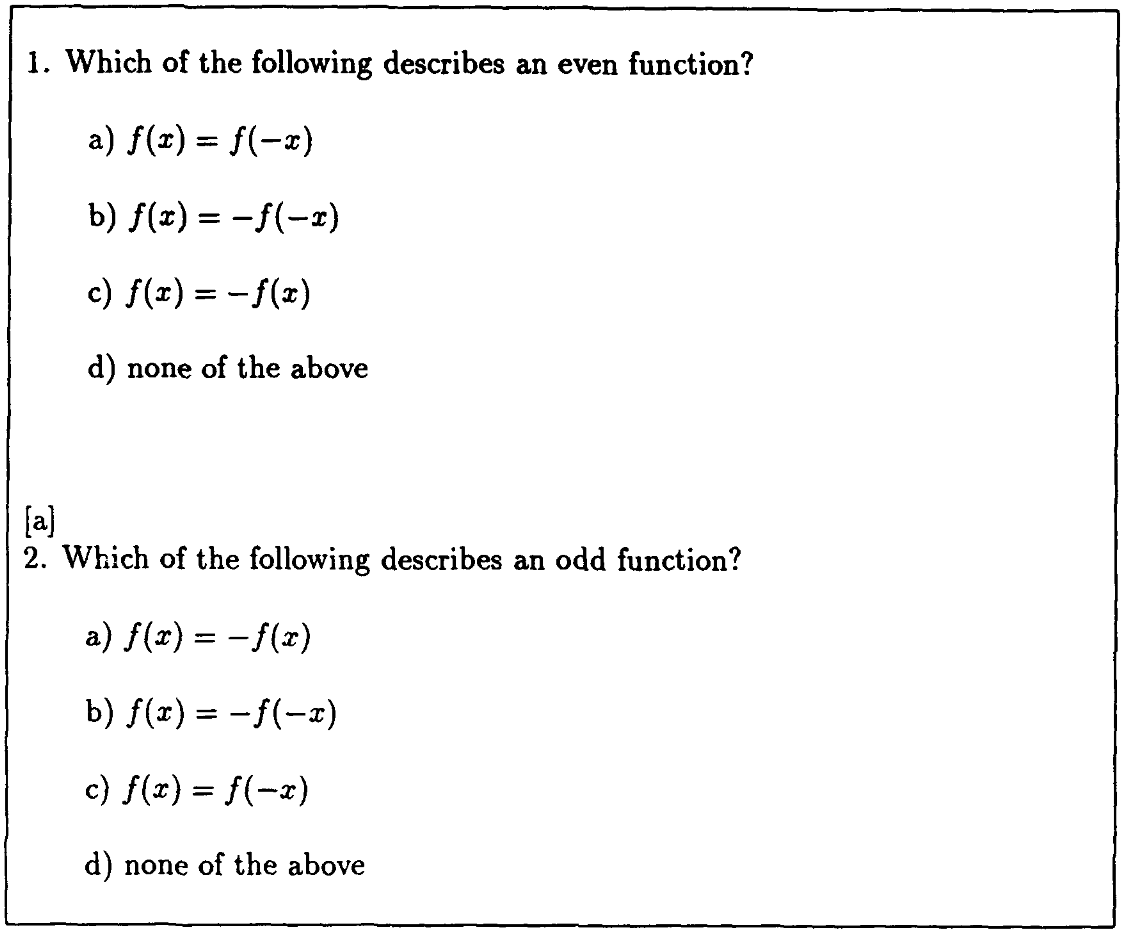
d.) D

*[b]

^ character is used to include a graph with this question.

(Program transfers control to a graph with four pictures on it.)

Figure 3.2



1. Which of the following describes an even function?

a) $f(x) = f(-x)$

b) $f(x) = -f(-x)$

c) $f(x) = -f(x)$

d) none of the above

[a]
2. Which of the following describes an odd function?

a) $f(x) = -f(x)$

b) $f(x) = -f(-x)$

c) $f(x) = f(-x)$

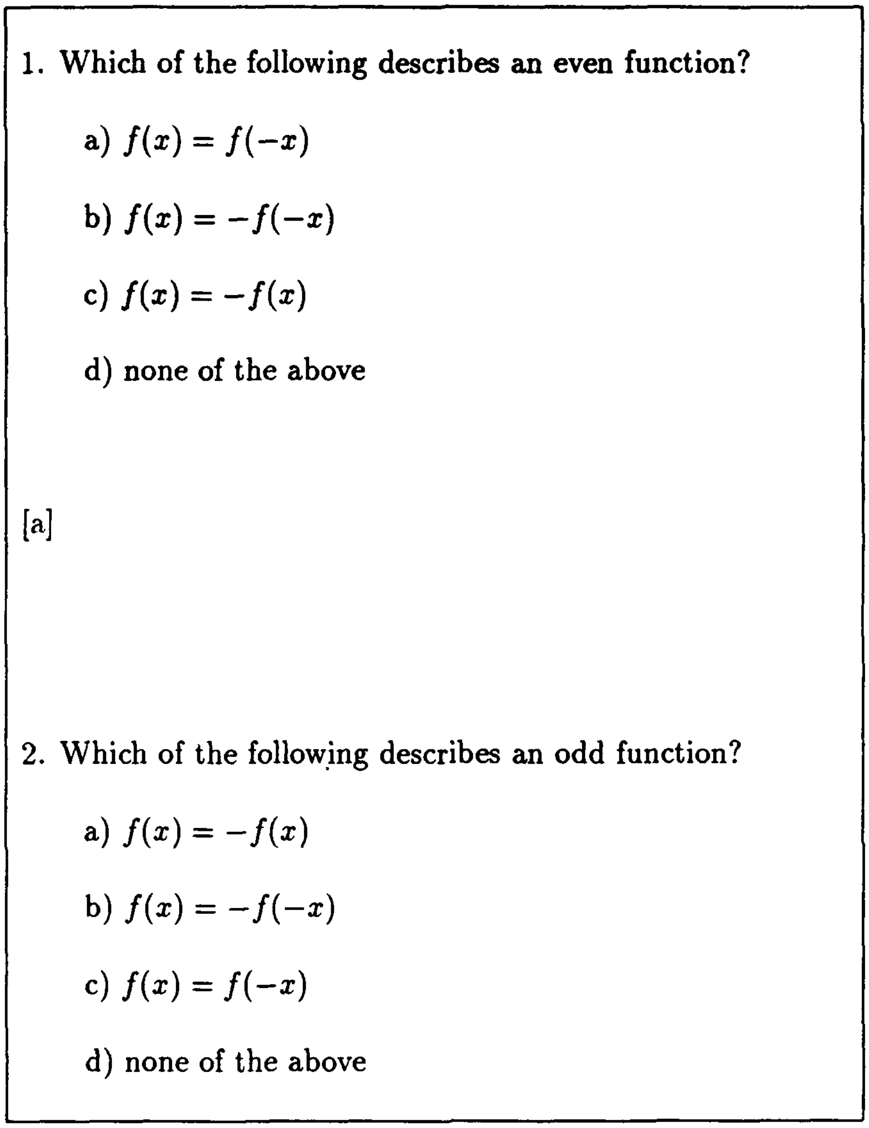
d) none of the above

(With this format, question 2 will begin in the upper left corner of the screen.)

Figure 3.3

Questions with graphs are indicated by the * appearing in the space just before the left bracket of the answer. Remember that the special characters *, [, and] must not appear anywhere else except to indicate the presence of a graph and to bracket an answer, respectively.

Before proceeding to Step 2, it is important to save a copy of the file `FINAL.DOC`. The copy file `FINAL.DOC` is slightly altered during its transformation into a refresher.



1. Which of the following describes an even function?

   a) $f(x) = f(-x)$

   b) $f(x) = -f(-x)$

   c) $f(x) = -f(x)$

   d) none of the above

[a]

2. Which of the following describes an odd function?

   a) $f(x) = -f(x)$

   b) $f(x) = -f(-x)$

   c) $f(x) = f(-x)$

   d) none of the above

(With this format, question 2 appears left justified and five lines down from the top of the screen. This is used to center the questions on the screen.)

Figure 3.4

Therefore, if any changes are to be made, adding questions for instance, then these changes can be made to the original file FINAL.DOC, and not the copy file that was transformed into the refresher.

STEP 2: Editing Step I.

The search and replace feature of the word processor is now used to place two spaces before and after the brackets of the actual answer in the file FINAL.DOC. This is needed to ensure that the program flows properly.

STEP 3: Editing Step II.

If the file FINAL.DOC was originally typed in document mode, extraneous carriage return/line feed pairs and other control characters may have been placed in the document. This is done to make the original text look neat, but is not necessary in the development of the refresher. Non-ASCII characters can be eliminated by invoking the following command:

```
STRIP FINAL.DOC
```

The program STRIP.COM removes these extraneous control characters. A new file is created, called FINAL.STR, which is then used to develop the refresher. An alternative method is to type the original document in the nondocument mode of the word processor. By using the nondocument mode, the extraneous control characters are not embedded in the file and thus Step 3 can be deleted.

The previous steps may seem to be very tedious in nature. Unfortunately, there is no simple way around this.

The following steps are the critical part of the programming to transform the file FINAL.DOC into the refresher and, fortunately, are very automated.

STEP 4: Preparing the questions.

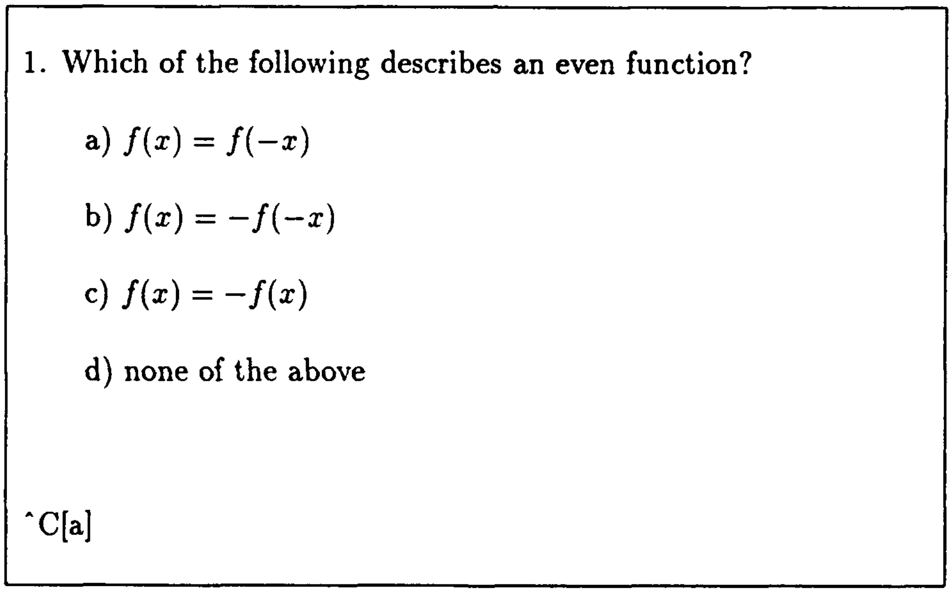
In order to display only the question and putative answers and not the actual answer, a control C (^C or 03h) is placed just before each left bracket of the actual answer. The program `CONTC.COM` is designed to perform this task. The following command is used:

```
CONTC FINAL.STR
```

or

```
CONTC FINAL.DOC
```

After the program `CONTC.COM` is run, the control C characters are placed in the space just prior to the left bracket of the actual answer in the file `FINAL.STR` (or the file `FINAL.DOC`) (see Figure 3.5).



1. Which of the following describes an even function?

   a) $f(x) = f(-x)$

   b) $f(x) = -f(-x)$

   c) $f(x) = -f(x)$

   d) none of the above

^C[a]

(File `FINAL.DOC` updated with control C placed before left bracket of correct answer.)

Figure 3.5

STEP 5: Blocking the output.

This step involves aligning the file FINAL.DOC in blocks of 128 bytes. This is accomplished by adding spaces after the right bracket of the correct answer so that each question/answer pair uses a multiple of 128 bytes. This is done in order to ease in the random selection of the questions. It is of the utmost importance that we know where each question begins and ends so that the refresher functions properly. The program ALIGN.EXE accomplishes this task. The program is invoked by using the following command:

```
ALIGN FINAL.STR
```

or

```
ALIGN FINAL.DOC
```

In a few seconds, a new file called FINAL.ALI is created. This file contains the questions and answers in aligned form that the refresher will use.

STEP 6: Separating question blocks.

The next step is to count how many 128 byte blocks are used in each question and to record these numbers. This data is used to determine where each question starts and how long it is. The programs COUNT.COM and HEXASC.COM are used for this purpose on the file FINAL.ALI. Type the following command:

```
COUNT FINAL.ALI
```

Before this program executes, the following message appears "Enter the hex word for the offset from earlier portions of the program (4 hex digits)." Respond by entering four zeros (0000).

```
DW 0000

DW 0002H,0004H,0006H,0008H,000AH,000BH,000CH,000EH

DW 000FH,0012H,0014H,0016H,0018H,001AH,001CH,001DH

DW 0020H,0021H,0025H,0028H,0029H,002AH,002BH,002CH

DW 002EH,0030H,0032H,0035H,0036H,0037H,003AH,003CH

DW 003DH,003EH,0040H,0043H,0046H,0048H,004AH,004CH

DW 004EH,0050H,0052H,0054H,0056H,0058H,005BH,005DH

DW 005FH,0061H,0063H,0065H,0067H,0069H,006CH,0070H

DW 0073H,0075H,0077H,0079H,007BH,007DH,007EH,0080H

DW 0082H,0084H,0086H,0088H,008AH,008CH,008EH,008FH

DW 0092H,0093H,0095H,0097H,0098H,009AH,009BH,009CH

DW 009EH,00A0H,00A2H,00A4H,00A6H,00A8H,00AAH,00ACH

DW 00ADH,00AFH,00B1H,00B3H,00B5H,00B7H,00B9H,00BAH

DW 00BBH,00BCH,00BEH,00C0H,00C2H,00C4H,00C6H,00C8H

DW 00CAH,00CCH,00CDH,00CFH,00D0H,00D2H
```

(file FINAL.AAA)

Figure 3.6

A new file called FINAL.CNT is created. This new file contains the data that is needed but in hex format. The refresher needs this information in ASCII format. To transform the data from hex format to ASCII format, the following command is used:

```
HEXASC FINAL.CNT
```

This creates the file FINAL.AAA. The file FINAL.AAA contains the data which will be used in the refresher (see Figure 3.6).

This data is used to determine the random access for the .ALI file. The hex words are the starting addresses of each question. The length of each question (in 128 long blocks) is the difference between consecutive entries. For example, question 3 starts in address 0004H and is two blocks long (the difference between 0006H and 0004H).

At this point in the procedure, the two files that will be used in the refresher are FINAL.ALI and FINAL.AAA.

STEP 7: Producing graphs.

This step is used to incorporate the graphs that will be used in the refresher. If the refresher contains no graphs, skip Step 7 and proceed to Step 8.The graphs used in the refresher have a white background with the actual graph either colored or black. There are two simple ways to accomplish this, however they are not the only ways.

The first way is to use the program *PCPAINT* with a mouse. By using the draw commands in *PCPAINT*, graphs are generated quickly and easily. The DOS graphics mode used by *PCPAINT* is mode 4 (320 pixels by 200 pixels). This may seem crude, however, it makes simple graphs that are quite useful.

A second way to generate graphs easily is by writing a basic program to draw the graphs on the screen. By using some of the following basic commands, color, line, set, pset, circle etc., simple graphs are quickly produced.

In order for the refresher to use the the graph generated above, a 'snapshot' of the graph must be taken from the screen. This is accomplished by the program RESIDENT.COM. This is a terminate and stay resident program. (A terminate and stay resident program is loaded into computer memory and terminates. It will not execute until a "hot key" is pressed to call it into action.) The "hot key" for RESIDENT.COM is the print screen key. Before the graph is generated on the screen, type the following command:

```
RESIDENT
```

This loads the program RESIDENT.COM in memory and it will run when the print screen key is pressed on the keyboard.

The next step is to generate the graph on the screen. When the desired graph is on the screen, press the print screen key. A new file, VIDRAM.DTA of length 16k, is created which contains the 'snapshot' of the graph on the screen. After VIDRAM.DTA is created, the system is rebooted to restore the original DOS pointers and to guarantee full DOS compatibility. After the reboot, VIDRAM.DTA is renamed to VID.000. If other graphs are desired for subsequent questions, the same procedure is followed. However, when renaming the 'snapshot' file VIDRAM.DTA, the file extension is changed to 001, 002 etc. As an example, if only questions 17, 38, 43 and 79 had graphs, then the graph for question 17 would be named VID.000. The graph for question 38 would be renamed from VIDRAM.DTA to VID.001, the graph for question 43 would be renamed from VIDRAM.DTA to VID.002 and the graph for question 79 would be renamed from VIDRAM.DTA to VID.003. The process of renaming graphs is crucial before the next 'snapshot' is taken to avoid writing over the file VIDRAM.DTA.

Since the majority of the graph consists of white background space, the files VID.*** can be compacted to save on disk space. The program DISPLAY.COM compactifies the 16K VID.*** files and renames them as PAK.*** files which are approximately 3K long. To accomplish this task, the following command is used:

```
DISPLAY VID.000
```

Repeat the process until all the VID.*** files are compacted to PAK.*** files. The refresher uses the compacted PAK.*** files in its presentation.

STEP 8: Organizing the process flow.

In this step, changes are made to the shell program. The shell program controls such things as the color that is displayed on the monitor, clearing the monitor after each question, displaying a new question, etc. The changes are necessary to accommodate the specifics of the refresher, such as total number of questions, the name of the file which contains the questions and answers, which question has a graph, etc. There are two versions of the shell program.

In the first version, the refresher prompts the user for a question number. This version is used mainly by the author of the refresher as a test to check that the questions are aligned properly on the screen and to ensure that the graphs are matched with the proper questions. The file TESTTEMP.ASM is the shell program used for this version.

The second version of the assembled refresher differs from the first in that a random integer generator is incorporated into the program. This is used to choose questions to appear on the screen in random order. It is this version that should be given to the students to sharpen their skills. The file TEMPLATE.ASM is the shell program used for this version.

The following information is needed to change the shell program:

- the total number of questions (from Step 1)
- the file FINAL.ALI (from Steps 2 thru 5)
- the file FINAL.AAA (from Step 6)

If graphs are used, the following is also required:

- the questions which have a graph (from Step 1)
- the total number of graphs
- the new control character for graphs, if * was not used.

The following changes are common to both shell programs, TESTTEMP.ASM and TEMPLATE.ASM. The changes are accomplished to the shell program with a word processor in the nondocument mode.

On line 46, replace MC2.ALI with the name of the .ALI file created in Step 5. In this example FINAL.ALI is placed on line 46 (see Figure 3.7).

On line 58, replace MC2.AAA with the .AAA file created in Step 6. In this example, FINAL.AAA is placed on line 58 (see Figure 3.8).

If graphs were used in the refresher, place the number of the question(s) which had a graph(s), on line 67. For example, if questions 17, 38, 43, and 79 had graphs, then change line 67 from DB 255 to DB 17, 38, 43, 79, 255 (255 is a number used by the program for proper flow). The corresponding PAK files are PAK.000, PAK.001, PAK.002, and PAK.003 (see Figure 3.9). (If question 17 needed three graphs, then 17 would appear three consecutive times on line 67 and there would be three corresponding *'s in question 17 in the aligned file, FINAL.ALI.)

On line 72 enter the total number of graphs that will be used in the refresher in decimal format. In the example above, a 4 is placed on line 72 (see Figure 3.10).

The last change that needs to be made is to change the message that initially appears on the screen, the INMSG. The INMSG starts on line 19. The message should contain pertinent information for the specific refresher, i.e. the course title, date, any particular instructions or messages, etc. The message is typed inside single quotation marks, a line at a time. Before the first quotation mark, type DB. After the last quotation mark, type a ,13,10. These are used for line feed and carriage return (see Figure 3.11).

In addition to the above changes, the following changes are made specifically to the file TESTTEMP.ASM. On line 154, input a number equal to 1 less than the number of questions in the refresher in hex format, not decimal format. For example, if the refresher had 90 questions, then 59H (59H = 89 decimal) and not 90, is placed on line 154 (see Figure 3.12). The number input on line 154 is one less than the actual number of questions because the questions start at 000 and not at 001.

line 46 in original file

```
;............................................
;in place of the MC2.ALI file, insert your FILE.ALI

FILEX     DB 'MC2.ALI',13

;............................................
```

line 46 in updated file

```
;............................................
;in place of the MC2.ALI file, insert your FILE.ALI

FILEX     DB 'FINAL.ALI', 13
;............................................
```

Figure 3.7

Line 58 in original file

```
;............................................
;in place of the MC2.AAA file, insert your file.AAA

INCLUDE MC2.AAA

;............................................
```

line 58 in updated file

```
;............................................
;in place of the MC2.AAA file, insert your file.AAA

INCLUDE FINAL.AAA ;.........................
```

Figure 3.8

line 67 in original file

```
;...............................................................
;Refer to Step 8. Place the number of the question which has a graph on
;line 67. Place them in ascending order. Leave 255 at the end of the list.

PICDAT DB 255
;...............................................................
```

line 67 in updated file

```
;...............................................................
;Refer to Step 8. Place the number of the question which has a graph on
;line 67. Place them in ascending order. Leave 255 at the end of the list.

PICDAT DB 17,38,43,79,255
;...............................................................
```

Figure 3.9

line 72 in original file

```
;.....................................................
;change the 0 to the number of graphs in use (not counting 255)

PICCNT DB    0         ;THE NUMBER OF PICTURES
;.....................................................
```

line 72 in updated file

```
;.....................................................
;change the 0 to the number of graphs in use (not counting 255)

PICCNT DB    4         ;THE NUMBER OF PICTURES
;.....................................................
```

Figure 3.10

line 19 in original file

```
;...............................................................
INMSG DB 'PRESENTATION QUESTIONS IN SELECTED ORDER',13,10,13,10
;
;        change INMSG as appropriate
;
;...............................................................
```

line 19 in updated file

```
;...............................................................
INMSG DB 'WELCOME TO THE CALCULUS REFRESHER',13,10
      DB 'VERSION 1.0    24 MAY 1990',13,10,13,10
;
;        change INMSG as appropriate
;
;...............................................................
```

Figure 3.11

```
line 154 in original file

;.......................................................

;Replace the FFh by the hex number of questions(less one). This
;number is initially set at its maximum (255).

    CMP AL,FFH         ;THE NUMBER OF QUESTIONS

;.......................................................

line 154 in updated file

;.......................................................

;Replace the FFh by the hex number of questions(less one). This
;number is initially set at its maximum (255).

    CMP AL,59H         ;THE NUMBER OF QUESTIONS

;.......................................................
```

Figure 3.12

If the control character for graphs was changed from * to \ for example, the following changes are necessary. On line 281, replace the * to \. On line 451, the same change is made (see Figure 3.13).

To change the opening message to reflect the number of questions in the refresher, line 33 must be updated. The number of questions in line 33 is one less than the actual number of questions. Enter the appropriate number as a three digit number, i.e., 089 rather than 89 if there are 90 questions used. It is one less because the first question is located in position 000 rather than at 001 (see Figure 3.14).

When these changes are made to TESTTEMP.ASM, save them and exit to DOS.

line 281 in original file

```
;.........................................................
;If flag for graphs is to change, replace * in line 281 with new
;flag such as \. see line 451 for a similar change.

DISP22:  CMP AL, '*'     ;CHECK IF QUESTION HAS A GRAPH
;.........................................................
```

line 281 in updated file

```
;.........................................................
;If flag for graphs is to change, replace * in line 281 with new
;flag such as \. See line 451 for a similar change.

DISP22:  CMP AL, '\'         ;CHECK IF QUESTION HAS A GRAPH
;.........................................................
```

line 452 in original file

```
;.........................................................
;If flag was changed in line 281, then make same change to line
;451 (replace * with \ for example)

    CMP AL,'*'         ;ARE THERE MORE GRAPH PAGES?
;.........................................................
```

line 452 in updated file

```
;.........................................................
;If flag was changed in line 281, then make same change to line
;451 (replace * with \ for example)

    CMP Al '\'         ;ARE THERE MORE GRAPH PAGES?
;.........................................................
```

Figure 3.13

```
line 33 in original file

;..........................................................

;Replace the 135 by one less than the LAST question number.

    DB '(000 THROUGH 135;THE NUMBERS DO NOT ALWAYS',13,10

;..........................................................

line 33 in updated file

;..........................................................

;Replace the 135 by one less than the LAST question number.

    DB '(000 THROUGH 089;THE NUMBERS DO NOT ALWAYS',13,10

;..........................................................
```

Figure 3.14

In addition to the initial changes, the following changes must be made specifically to the file TEMPLATE.ASM.

On line 196 and line 198, input the actual number of questions in the refresher in hex format, not decimal format. For example, if the refresher had 104 questions, then 68H is placed on lines 196 and 198. These lines update the the random number generator for version two of the refresher (see Figure 3.15).

If the control character for the graphs was changed from * to \ for example, the following changes are also necessary. On line 274 change * to \. On line 446, the same change is made (see Figure 3.13).

When these changes are made to TEMPLATE.ASM, save them and return to DOS.

lines 196-198 in original file

```
;..................................................
;In two places, change FFh to the hex number of questions.
;This is the random number generator.
GET1:    CMP AL,FFH
         JBE EXIT1
         SUB AL,FFH
;..................................................
```

lines 196-198 in updated file

```
;..................................................
;In two places, change FFh to the hex number of questions.
;This is the random number generator.
GET1:    CMP AL,68H
         JBE EXIT1
         SUB AL,68H
;..................................................
```

Figure 3.15

STEP 9: Final assembly.

The final step in producing the refresher is to assemble it into an executable file. Before any commands are given to assemble the refresher, make sure that the shell program (TESTTEMP.ASM or TEMPLATE.ASM), the files FINAL.ALI and FINAL.AAA, the PAK.*** files and MASM.EXE are in the same directory. To assemble the refresher the following command is input:

```
MASM TESTTEMP
```

or

```
MASM TEMPLATE
```

While MASM.EXE is assembling the program, it will prompt for additional file names. A carriage return at each prompt will suffice. After MASM.EXE is finished, the following command is typed:

```
LINK TESTTEMP
```

or

```
LINK TEMPLATE
```

Link will also prompt for file names. Again, carriage returns will suffice. After link is through executing, the executable file is created, TESTTEMP.EXE or TEMPLATE.EXE. At this point it is a good idea to rename the executable file to a file name which corresponds to the name of the course the refresher was written for. This is accomplished by using the DOS command RENAME. As an example,

```
RENAME TEMPLATE.EXE MA1117.EXE
```

will change the name from TEMPLATE.EXE to MA1117.EXE. Now by typing MA1117 followed by a carriage return, the refresher begins.

IV. SUMMARY

The following outline is meant to be used as a reference when developing a refresher.

- DEVELOP FILE OF QUESTIONS AND ANSWERS (FINAL.DOC)
- STRIP FINAL.DOC (IF NECESSARY)
- CONTC FINAL.DOC
- ALIGN FINAL.DOC (PRODUCES FINAL.ALI)
- COUNT FINAL.ALI (PRODUCES FINAL.CNT)
- HEXASC FINAL.CNT (PRODUCES FINAL.AAA)
- UPDATE TEMPLATE.ASM OR TESTTEMP AS NECESSARY
- MASM TESTTEMP OR MASM TEMPLATE
- LINK TESTTEMP OR LINK TEMPLATE

If graphs are used, see Step 7 for instructions.

V. OPERATING INSTRUCTIONS

The following files are needed to be in the same directory for the refresher to function properly:

- the executable file (MA1117.EXE)
- the aligned file (FINAL.ALI)
- the PAK.*** files

The refresher is started by typing the file name (MA1117). After the opening message appears, press any key to proceed to the first question. When the question is answered, press any key and the correct answer appears in red under the question. To proceed to the next question, press any key. If a graph appears, press control Q to toggle back to the question or control A to toggle to the answer. If control A is pressed, press any key for the correct answer to appear, then any other key for the next question. To end the refresher, press control C at any time.

VI. MISCELLANEOUS INSTRUCTIONS

Additionally, the following instructions may be of some use in developing the refresher.

- If a question is more than one screenful in length, then the question is split up using the 'MORE' command. In the file FINAL.ALI, change any unimportant character to 02 (^b). This will cause the display to pause at that spot until any key is pressed. To enter the 02 (^b), debug is used as most word processors will not be up to the task. Otherwise, stick to a screenful at a time.

- If graphs are used in the refresher then the following programs may be used to assist in developing the the graphs. The programs are used to view a graph outside of the refresher environment.

  - MODE3.COM
  - MODE4.COM
  - UDISPLAY.COM
  - CHNGCOLS.COM

The graphs in the refresher are in mode 4. In mode 4, the screen is manipulated pixel by pixel. When the computer initializes itself, it sets the mode to mode 3, alpha-numeric. To change the mode of the computer to mode 4 so that a graph can be seen outside the refresher, type

```
MODE4
```

The program MODE4.COM changes the mode from mode 3 to mode 4. By typing

```
UDISPLAY PAK.000
```

the PAK.000 file will be displayed on the screen. To place the terminal into mode 3, type

```
MODE3
```

The terminal will return to its original mode, mode 3.

- The program CHNGCOLS.COM is used to change any specific color to any other specific color in mode 4. By typing

  ```
  CHNGCOLS
  ```

  the program prompts for color changes for use in the graphs.

- When typing the questions and proposed answers along with the correct answer, the extended ASCII character set may be used. These are input by holding the Alt Key and typing in the corresponding ASCII code. For example, if the mathematical constant pi is to be displayed, then by holding the Alt Key and typing 227, the symbol π appears.

- If more than one refresher is to be placed on a diskette, then overlap of the PAK.*** files will occur. This overlap occurs because the graph PAK.000 will be the first graph that will appear in each refresher on the diskette. To alleviate this problem, rename all the PAK.*** files that belong to a particular refresher to DAK.*** files. (This renaming is anything appropriate, such as CAK.*** or HAK.***). In addition, the shell program needs to be updated to reflect this change. In the shell TESTTEMP.ASM, change the P on line 213 to a D. In the shell program TEMPLATE.ASM, this change occurs on line 206.

VII. CONCLUSION

The main benefit of the refresher is for students to experience faster learning and greater retention of course objectives. Observations have been made which indicate that the use of refreshers have accomplished this goal. Professor Gordon Latta of the Naval Post Graduate School has collected data on medical students which confirm this. On average, medical students would take a difficult medical exam ten times before passing it. By allowing students to study from refreshers developed similarly to the calculus refresher, the medical students passed the exam on their first try. Similar statistics should be kept on students who use the calculus refresher.

Within the framework of the refresher, other uses may be developed. As an example, each screen may be copied onto slides. These can be used to highlight specific points of interest instead of questions and answers. By paging through the screens in sequential order using the `TESTTEMP.ASM` shell, presentations can be made. Another use is for organizing and storing lecture notes and lesson plans for a class. If a new instructor teaches a class, he may refer to notes recorded by a previous instructor using this method to help him organize and teach the class more effectively.

With slight modifications, the questions that appear in random order can also be printed. Hence, a test bank of questions and answers for exams could be developed. If an instructor wanted to give a practice exam or a validation exam, he could ask the refresher to print 40 or 50 questions at random from the test bank. This could then constitute the validation exam. In this way, time could be saved by the instructor in preparing and grading an exam and each student would receive a different exam. New questions could easily be added and old questions deleted to keep the test bank current.

Following the framework described, many refreshers (on a variety of course objectives) may be developed much to the benefit of students in the armed forces.

APPENDIX A

```
CODE SEGMENT PARA PUBLIC 'CODE'
FCB EQU 005CH
DTA EQU 0080H
OPENF EQU 0FH ;OPEN FILE REFERENCED IN THE FCB
CLOSEF EQU 10H ;CLOSE FILE
SRCHFRST EQU 11H ;SEARCH FOR FIRST OCCURRENCE
SRCHNEXT EQU 12H ;SEARCH FOR NEXT OCCURRENCE
DELETEF EQU 13H ;DELETE FILE
READS EQU 14H ;READ SEQUENTIALLY
WRITES EQU 15H ;WRITE SEQUENTIALLY
MAKEF EQU 16H ;MAKE FILE
SETDMA EQU 1AH ;SET DISK TRANSFER ADDRESS
PARSE EQU 29H ;PARSE FILENAME,  SEE PG 5-71 TECH.REF
SELDSK EQU 0EH ;SELECT DRIVE
ORG 0100H
START PROC FAR
ASSUME CS:CODE
ASSUME ES:CODE ;ES POINTS TO OUR PROG. SEGMENT
ASSUME DS:CODE ;NOW POINT DS TO OUR SEGMENT
CALL CRLF
MOV DX,OFFSET INMSG
MOV AH,9
CALL BDOS
CALL CRLF
MOV BX,OFFSET DTA
MOV AL,[BX]
OR AL,AL
JNZ BEG
JMP ERR3
BEG: MOV DX, FCB
MOV AH,SRCHFRST
CALL BDOS
OR AL,AL ;00 ==>MATCHING FILENAME FOUND
JZ BEG1
JMP FNFERR
BEG1: MOV DX, FCB
MOV AH,OPENF
CALL BDOS ;OPEN THE FILE
MOV DX,OFFSET BUFFER
```

```
BEG2: PUSH DX
MOV AH,SETDMA
CALL BDOS
MOV AH,READS
MOV DX, FCB
CALL BDOS ;READ A SECTOR
OR AL,AL
JZ BEG3
JMP ENTER ;WRITE BACK TO DISK
BEG3: POP DX
MOV SI,80H
ADD DX,SI
JMP BEG2 ;LOOP UNTIL EOF
ENTER: POP DX
ADD DX,80H
MOV BX,OFFSET BUFFER
SUB DX,BX
MOV CX,DX ;COUNT IS IN CX
MOV BX,OFFSET BUFFER
ENTER1: MOV AL,[BX]
AND AL,7FH ;STRIP NOW
MOV [BX],AL
INC BX
DEC CX
JNZ ENTER1
MOV BX,FCB
MOV CX,WORD PTR 16[BX]
MOV DX,WORD PTR 18[BX] ;THE BYTE COUNT
MOV BX,CX
MOV CX,7
SHR BX,CL
MOV CX,9
SHL DX,CL
ADD BX,DX ;BX HAS THE NUMBER OF RECORDS
INC BX ;TO ACCOUNT FOR FRACTIONS
MOV CX,BX ;THE COUNT IN RECORDS
MOV BX,FCB
MOV BYTE PTR 9[BX],'S'
MOV BYTE PTR 10[BX],'T'
MOV BYTE PTR 11[BX],'R' ;NEW FILE TYPE IS STR
MOV BYTE PTR 32[BX],0 ;RESET THE FILE
MOV AH,DELETEF
MOV DX,FCB
CALL BDOS ;REMOVE ANY EARLIER VERSIONS
MOV AH,MAKEF
MOV DX,FCB
CALL BDOS ;CREATE THE FILE
MOV DX,OFFSET BUFFER
ELOOP: PUSH DX
MOV AH,SETDMA
CALL BDOS
MOV DX,FCB
MOV AH,WRITES
CALL BDOS
POP DX
MOV SI,80H
ADD DX,SI
DEC CX
JZ ELUP
JMP ELOOP
ELUP: MOV AH,CLOSEF
MOV DX,FCB
CALL BDOS
INT 20H ;FAR RETURN, ALL DONE
FNFERR: MOV DX,OFFSET FNFMSG
MOV AH,9
CALL BDOS ;REPORT FILE NOT FOUND
CALL CRLF
INT 20H ;FAR RETURN
ERR3: MOV DX,OFFSET ERR3MSG
MOV AH,9
CALL BDOS
INT 20H ;FAR RETURN
CRLF PROC NEAR
    PUSH DX
    PUSH AX
    MOV DL,0DH
    MOV AH,02H
    CALL BDOS        ;DO A <CR> AND <LF>
    MOV DL,0AH
    MOV AH,02H       ;SAVING MOST REGISTERS
    CALL BDOS
    POP AX
    POP DX
    RET
CRLF ENDP
BDOS PROC NEAR
PUSH SI
     PUSH ES
     PUSH DX
     PUSH CX
     PUSH BX
    INT 21H
    POP BX
    POP CX
    POP DX
    POP ES
POP SI
    RET
BDOS ENDP
INMSG DB 'THIS PROGRAM STRIPS THE HIGH BIT FROM ASCII FILES',13,10
DB ' AND CREATES A NEW FILE WITH THE TYPE .STR',13,10
DB 'ENTER ANY KEY TO CONTINUE',13,10,'$'
FNFMSG DB 'FILE NOT FOUND, RETURNING TO DOS',13,10,'$'
ERR3MSG DB 'NO PARAMETERS ENTERED, RETURNING TO DOS',13,10
DB 'THE CORRECT FORMAT IS',13,10
DB 'STRIP FILE.NAM',13,10,13,10,'$'
BUFFER EQU $
START ENDP
CODE ENDS
END START
```

APPENDIX B

```
CODE SEGMENT PARA PUBLIC 'CODE'
ORG 0100H
ASSUME CS:CODE, ES:CODE,DS:CODE
START: JMP BEGIN
DTA EQU 80H
FCB EQU 005CH
OPENF DB 0FH ;OPEN FILE REFERENCED IN THE FCB
CLOSEF DB 10H ;CLOSE FILE
SRCHFRST DB 11H ;SEARCH FOR FIRST OCCURRENCE
SRCHNEXT DB 12H ;SEARCH FOR NEXT OCCURRENCE
DELETEF DB 13H ;DELETE FILE
READS DB 14H ;READ SEQUENTIALLY
WRITES DB 15H ;WRITE SEQUENTIALLY
MAKEF DB 16H ;MAKE FILE
SETDMA DB 1AH ;SET DISK TRANSFER ADDRESS
PARSE DB 29H ;PARSE FILENAME,  SEE PG 5-71 TECH.REF
SELDSK DB 0EH ;SELECT DRIVE
INMSG DB 'INSERTS CONTROL C CHARACTERS TO MARK ANSWERS',13,10
DB 'ENTER ANY KEY TO CONTINUE',13,10,'$'
FNFMSG DB 'FILE NOT FOUND, RETURNING TO DOS',13,10,'$'
ERR3MSG DB 'NO PARAMETERS ENTERED, RETURNING TO DOS',13,10
DB 'THE CORRECT FORMAT IS',13,10
DB 'CONTC FILE.NAM',13,10,13,10,'$'
LSAVE DW 00H
ADDR DB 4 DUP(0)
BEGIN: MOV DX,OFFSET INMSG
MOV AH,9
CALL BDOS
CALL CRLF
MOV AH,1
CALL BDOS
MOV BX, DTA
MOV AL,[BX]
OR AL,AL
JNZ BEG
JMP ERR3
BEG: MOV DX, FCB
MOV AH,SRCHFRST
CALL BDOS
```

```
OR AL,AL ;00 ==>MATCHING FILENAME FOUND
JZ BEG1
JMP FNFERR
BEG1: MOV DX, FCB
MOV AH,OPENF
CALL BDOS ;OPEN THE FILE
MOV DX,OFFSET BUFFER
BEG2: MOV AH,SETDMA
INT 21H ;OPEN THE BUFFER FOR DATA TRANSFER
PUSH DX
MOV AH,READS
MOV DX, FCB
CALL BDOS ;READ A SECTOR
CMP AL,1
JNZ BEG3 ;1==>EOF
POP DX
ADD DX,80H
MOV AX,OFFSET BUFFER
SUB DX,AX
INC DX
MOV CX,DX
MOV WORD PTR LSAVE,DX ;SAVE FILE LENGHT
MOV BX,OFFSET BUFFER
BEG4: MOV AL,[BX]
CMP AL,5BH
JNZ BEG42
JMP FIXIT
BEG42: INC BX
DEC CX
BEG41: JNZ BEG4
MOV BX,005CH
MOV AL,0
MOV 12[BX],AL
MOV 13[BX],AL
MOV 32[BX],AL ;ZERO OUT CURRENT RECORD AND BLOCK
MOV CX,WORD PTR LSAVE
MOV DX,OFFSET BUFFER
LAST: MOV AH,SETDMA
INT 21H ;PREPARE TO WRITE THE FILE BACK
PUSH DX
MOV DX,FCB
MOV AH,WRITES
```

```
INT 21H
POP DX
ADD DX,80H
SUB CX,80H
JGE LAST
POP AX
MOV AH,CLOSEF
MOV DX,FCB
INT 21H ;CLOSE THE FILE AND GO HOME
INT 20H ; ALL DONE
BEG3: POP DX
ADD DX,80h
JMP BEG2 ;LOOP UNTIL EOF
FIXIT: DEC BX
MOV AL,03H
MOV [BX],AL
INC BX
INC BX
DEC CX
JMP BEG41
FNFERR: MOV DX,OFFSET FNFMSG
MOV AH,9
CALL BDOS ;REPORT FILE NOT FOUND
CALL CRLF
RET ;FAR RETURN
ERR3: MOV DX,OFFSET ERR3MSG
MOV AH,9
CALL BDOS
RET ;FAR RETURN
CRLF PROC NEAR
    PUSH DX
    PUSH AX
    MOV DL,0DH
    MOV AH,02H
    CALL BDOS       ;DO A <CR> AND <LF>
    MOV DL,0AH
    MOV AH,02H      ;SAVING MOST REGISTERS
    CALL BDOS
    POP AX
    POP DX
    RET
CRLF ENDP
```

```
BDOS PROC NEAR
PUSH SI
     PUSH ES
     PUSH DX
     PUSH CX
     PUSH BX
    INT 21H
    POP BX
    POP CX
    POP DX
    POP ES
POP SI
    RET
BDOS ENDP
BUFFER EQU $
START ENDP
CODE ENDS
END START
```

APPENDIX C

```
STACK SEGMENT PARA STACK 'STACK'
DB 256 DUP(0)
STACK ENDS
DATA SEGMENT PARA PUBLIC 'DATA'
DSKMSG   DB   'DISK FULL, RETURNING TO DOS$'
ADDR     DB   4 DUP(0)
SIGNON   DB   'FILE ALIGNMENT PROGRAM VERS. 1.1',10,13
         DB   'READS MEDICAL ASCII FILES AND ',10,13
         DB   'PADS TO A MULTIPLE OF 128 BYTES FOR',10,13
         DB   'RANDOM ACCESS USE LATER.',10,13,'$'
CORRECTION    DB   'THE CORRECT COMMAND IS',0DH,0AH
         DB   'ALIGN FILE.NAM',0DH,0AH
         DB   'START OVER$'
INERR1  DB    'INPUT ERROR, START OVER$'
BUFFER   DB      0080H DUP(0)
 DB     00H,00H
FCB      DB 37 DUP(0)
FCB1     DB 37 DUP(0)
OPENF    EQU    15
SELDSK   EQU    14
SRCHFRST EQU    17
MAKEF    EQU    22
READS    EQU    20
SETDMA   EQU    26
WRITES   EQU    21
CLOSEF   EQU    16
DELETEF  EQU    19
COUNT     DB     0
CHAR      DB     0AH ;DENOTES END OF ANSWER
BIGBUF   DB 0F400H DUP(0)
DATA ENDS
CODE SEGMENT PARA PUBLIC 'CODE'
START PROC FAR
ASSUME CS:CODE
PUSH DS
MOV AX,0
PUSH AX ;RETURN ADDRESS TO DOS
MOV AX,DATA
MOV ES,AX
```

```
ASSUME ES:DATA ;POINT ES TO DATA SEGMENT FOR PARMS TRANSFER
MOV SI,5CH
MOV DI,OFFSET FCB
CLD
MOV CX,12
REP MOVSB ;FIRST,GET FILE NAME
MOV SI,0080H
MOV DI,OFFSET BUFFER
MOV CX,128
REP MOVSB ;THEN GET THE DEFAULT DMA BLOCK
MOV DS,AX
ASSUME DS:DATA ;NOW GO TO OUR OWN DATA SEGMENT
BEG:    MOV DX,OFFSET SIGNON
        CALL PRINT
    CALL CRLF
    MOV BX,OFFSET BUFFER
    MOV AL,[BX]
    OR AL,AL
    JNZ BEG1 ;0=NO FILE NAME ENTERED AT ALL
    MOV DX,OFFSET CORRECTION
    CALL PRINT
    JMP EXIT1 ;REPORT THE OMISSION AND QUIT
BEG1:    CALL CRLF
BEG3:    MOV BX,OFFSET FCB
    MOV DI,OFFSET FCB1
    MOV CL,08H
BEG3LUP: INC BX
    INC DI
    MOV AL,[BX]
    MOV [DI],AL
    DEC CL
    JNZ BEG3LUP  ;COPY FILE NAME TO FCB1
    INC DI
    MOV AL,'A'
    MOV [DI],AL
    INC DI
    MOV BYTE PTR [DI],'L'
    INC DI
    MOV BYTE PTR [DI],'I'
    MOV AH,DELETEF ;DELETE ANY FILE WITH SAME ENTRY
    MOV DX,OFFSET FCB1
    CALL BDOS
    MOV AH,OPENF ;OPEN THE FILES
    MOV DX, OFFSET FCB
    CALL BDOS
    MOV AH,MAKEF ;OR MAKE THEM AS APPROPRIATE
    MOV DX,OFFSET FCB1
    CALL BDOS ;OPEN BOTH FILES
MOV DX,OFFSET BUFFER
MOV AH,SETDMA
CALL BDOS ;SET DMAADD FOR DISK TRANSFER
    MOV DI,OFFSET BIGBUF     ;FOR THE REPLACE FILE
RLUP:    MOV BX,OFFSET BUFFER
RLUP0: CALL FLUP ;READ IN ONE SECTOR TO DMAADD
       MOV CH,80H
RLUP1:   MOV AL,[BX]
    CMP AL,03H
    JNZ RLUP2
    JMP FIXIT2
    JMP RLUP3 ;TRANSFER TO BIGBUF, REPLACING AS WE GO
RLUP2:   MOV [DI],AL
RLUP3:  INC BX
    INC DI
    DEC CH
    JNZ RLUP1 ;BY HERE, THE SECTOR IS NOW IN BIGBUF
    JMP RLUP
WLUP:    MOV AL,1AH
    MOV [DI],AL
    INC DI
    MOV [DI],AL ;APPEND TWO ^Z=1AH FOR EOF
    MOV BX,OFFSET BIGBUF
    SUB DI,BX ;DI NOW EQUALS LENGTH OF FILE IN BYTES
    MOV CL,7
    SHR DI,CL  ;DIVIDE BY 128
    INC DI ;DI NOW EQUALS THE NUMBER OF SECTORS
               ;IN THE FILE, TO BE WRITTEN
    PUSH DI        ;SAVE THE RECORD COUNT
    MOV AH,SETDMA
    MOV DX,OFFSET BIGBUF
    CALL BDOS      ;NEW DMAADD FOR FILE TRANSFER
    POP DI
    MOV SI,OFFSET BIGBUF     ;POINTER FOR DMAADD
    PUSH SI
WLUP1:   MOV DX,OFFSET FCB1
```

```
    MOV AH,WRITES
    PUSH DI         ;SAVE COUNT
    CALL BDOS
    POP DI
    OR AL,AL
    JZ WLUP2
    JMP DISKFULL
WLUP2:   POP SI
    ADD SI,80H
    MOV DX,SI
    PUSH SI
    PUSH DI
    MOV AH,SETDMA ;INCREMENT THE DTA THROUGH BIGBUF
    CALL BDOS
    POP DI
    DEC DI
    JNZ WLUP1
POP SI
    MOV AH,CLOSEF
    MOV DX,OFFSET FCB1
    CALL BDOS ;FILE TRANSFERED AND CLOSED
RET ;RETURN TO DOS, TRANSFER COMPLETED
FIXIT2:  MOV AL,[BX]
    CMP AL,CHAR
    JNZ FIXIT1
FIXIT21:    PUSH CX
    CALL SPACFILL
FIXIT211: MOV [DI],AL
    MOV AL,' '
    INC DI
    DEC CH
    JNZ FIXIT211
    POP CX
    INC BX ;POINT PAST THE  LF
    DEC CH
    JNZ FIXIT212
    JMP RLUP
FIXIT212:    JMP RLUP1
FIXIT1:  MOV [DI],AL
    INC BX
    INC DI
    DEC CH
    JNZ FIXIT2
MOV BX,OFFSET BUFFER
CALL FLUP
MOV CH,80H
JMP FIXIT2
SPACFILL PROC NEAR
    PUSH AX
    PUSH BX
    MOV BX,OFFSET BIGBUF
    MOV AX,DI
    SUB AX,BX
    AND AL,7FH
    NOT AL
    INC AL
    AND AL,7FH
    MOV CH,AL
    POP BX
    POP AX
    RET
SPACFILL ENDP
FLUP PROC NEAR
    PUSH DI
    MOV AH,READS ;ROUTINE TO READ ONE SECTOR TO DTA
    MOV DX,OFFSET FCB
    CALL BDOS
    CMP AL,1
    JZ EXIT
    POP DI
    RET
EXIT:    POP DI
POP AX ;FIX UP THE STACK
    JMP WLUP
FLUP ENDP
CRLF    PROC NEAR
PUSH DX
PUSH CX
PUSH BX
PUSH AX
    MOV DL,0DH
    MOV AH,02H
    INT 21H  ;DO A <CR> AND <LF>
    MOV DL,0AH
    MOV AH,02H
    INT 21H
    POP AX
POP BX
POP CX
POP DX
    RET
CRLF ENDP
CONOUT  PROC NEAR ;CONSOLE OUTPUT WITH PAUSE
PUSH DX
PUSH CX
PUSH BX
PUSH AX
    MOV AH,06H     ;TEST FOR INPUT
    MOV DL,0FFH
    INT 21H
    OR AL,AL  ;AL=1 IF KEY PRESSED, ELSE 0
    JNZ PAUSE
OUT1:    POP AX
PUSH AX
    MOV DL,AL
    MOV AH,02H
    INT 21H
    POP AX
POP BX
POP CX
POP DX
    RET
PAUSE:   MOV AH,06H
    MOV DL,0FFH
    INT 21H
    OR AL,AL
    JZ PAUSE
    JMP OUT1
CONOUT ENDP
PRINT   PROC NEAR
PUSH AX
PUSH BX
PUSH CX
PUSH DX
    MOV AH,09H
    INT 21H  ;CALL HERE WITH DX=OFFSET MESSAGE
    POP DX
POP CX
POP BX
POP AX
RET
PRINT ENDP
DISKFULL:     POP SI
    MOV DX,OFFSET DSKMSG
    CALL PRINT
    JMP EXIT1
BDOS PROC NEAR
PUSH BX
PUSH CX
PUSH DX
PUSH ES
    INT 21H
    POP ES
POP DX
POP CX
POP BX
    RET
BDOS ENDP
EXIT1: RET ;FAR RETURN, HOPEFULLY
START ENDP
CODE ENDS
END START
```

APPENDIX D

```
CODE SEGMENT PARA PUBLIC 'CODE'
FCB EQU 005CH
DTA EQU 0080H
OPENF EQU 0FH ;OPEN FILE REFERENCED IN THE FCB
CLOSEF EQU 10H ;CLOSE FILE
SRCHFRST EQU 11H ;SEARCH FOR FIRST OCCURRENCE
SRCHNEXT EQU 12H ;SEARCH FOR NEXT OCCURRENCE
DELETEF EQU 13H ;DELETE FILE
READS EQU 14H ;READ SEQUENTIALLY
WRITES EQU 15H ;WRITE SEQUENTIALLY
MAKEF EQU 16H ;MAKE FILE
SETDMA EQU 1AH ;SET DISK TRANSFER ADDRESS
PARSE EQU 29H ;PARSE FILENAME,  SEE PG 5-71 TECH.REF
SELDSK EQU 0EH ;SELECT DRIVE
ORG 0100H
START PROC FAR
ASSUME CS:CODE
ASSUME ES:CODE ;ES POINTS TO OUR PROG. SEGMENT
ASSUME DS:CODE ;NOW POINT DS TO OUR SEGMENT
CALL CRLF
MOV DX,OFFSET INMSG
MOV AH,9
CALL BDOS
CALL CRLF
MOV DX,OFFSET MSG1
MOV AH,9
CALL BDOS
CALL CRLF
CALL IN4
CALL CONV4
MOV WORD PTR SHIFT,DX ;SAVE THE SHIFT OFFSET
MOV BX,OFFSET DTA
MOV AL,[BX]
OR AL,AL
JNZ BEG
JMP ERR3
BEG: MOV DX, FCB
MOV AH,SRCHFRST
CALL BDOS
```

```
OR AL,AL ;00 ==>MATCHING FILENAME FOUND
JZ BEG1
JMP FNFERR
BEG1: MOV DX, FCB
MOV AH,OPENF
CALL BDOS ;OPEN THE FILE
MOV DX,OFFSET BUFFER
BEG2: PUSH DX
MOV AH,SETDMA
CALL BDOS ;OPEN BUFFER FOR DMA
MOV AH,READS
MOV DX, FCB
CALL BDOS ;READ A SECTOR
CMP AL,1
JNZ BEG3
JMP ENTER ;DONE LOADING, NOW COUNT RECORDS
BEG3: POP DX
MOV SI,80H
ADD DX,SI
JMP BEG2 ;LOOP UNTIL EOF
ENTER: POP DX
ADD DX,80H ;WHOLE FILE NOW INCLUDED
MOV BX,OFFSET BUFFER ;THE FILE BEGINNING
SUB DX,BX ;FILE LENGTH IN DX
MOV SI,OFFSET CNTBUF
ENTER2: MOV AL,[BX] ;READ THE BUFFER FOR CNTL-C
CMP AL,03H
JZ ENTER1
INC BX
DEC DX
JNZ ENTER2
JMP ENTER21
ENTER1: MOV AL,[BX]
CMP AL,0AH
JZ ENTER11
INC BX
JMP ENTER1
ENTER11: MOV AX,BX
SUB AX,OFFSET BUFFER
MOV CX,7
SHR AX,CL
INC AX
```

```
ADD AX,WORD PTR SHIFT
MOV [SI],AX
INC SI
INC SI
INC BX
JMP ENTER2
ENTER21: MOV AX,OFFSET CNTBUF
SUB SI,AX ;LENGTH OF COUNT BUFFER
MOV CX,SI
MOV BX,FCB
MOV AL,'C'
MOV 9[BX],AL
MOV AL,'N'
MOV 10[BX],AL
MOV AL,'T'
MOV 11[BX],AL
MOV AL,0
MOV 32[BX],AL
MOV 12[BX],AL
MOV 13[BX],AL   ;REMOVE ANY EARLIER VERSIONS
MOV AH,MAKEF
MOV DX,FCB
CALL BDOS ;CREATE THE FILE
MOV DX,FCB
MOV AH,OPENF
INT 21H
MOV DX,OFFSET CNTBUF
ELOOP: PUSH DX
MOV AH,SETDMA
CALL BDOS
MOV DX,FCB
MOV AH,WRITES
CALL BDOS
POP DX
MOV SI,80H
ADD DX,SI
SUB CX,80H
JNS ELOOP
ELUP: MOV AH,CLOSEF
MOV DX,FCB
CALL BDOS
INT 20H ;FAR RETURN, ALL DONE
```

```
FNFERR: MOV DX,OFFSET FNFMSG
MOV AH,9
CALL BDOS ;REPORT FILE NOT FOUND
CALL CRLF
INT 20H ;FAR RETURN
ERR3: MOV DX,OFFSET ERR3MSG
MOV AH,9
CALL BDOS
INT 20H ;FAR RETURN
ZCAH   PROC NEAR
    SUB AL,30H     ;CONVERT ASCII BYTE IN AL TO HEX DIGIT IN AL
    JB ERR2
    CMP AL,0AH
    JNB ZCAH1 ;ON RETURN, 20H = ERROR CONDITION
    RET
ZCAH1:   SUB AL,07H     ;USES AL ONLY
    CMP AL,0AH
    JB ERR2
    CMP AL,10H
    JNB ERR2
    RET
ERR2:    MOV AL,20H
    RET
ZCAH ENDP
ZCHA  PROC NEAR
    AND AL,0FH     ;CONVERT HEX DIGIT IN AL TO ASCII BYTE IN AL
    ADD AL,90H
    DAA
    ADC AL,40H     ;DIGIT IS IN LOW NYBBLE
    DAA
    RET       ;OPTIMIZED SUBROUTINE
ZCHA ENDP
ZEN   PROC NEAR
     MOV CL,04H     ;EXCHANGE NYBBLES
    ROL AL,CL
    RET       ;USES CL,AL
ZEN ENDP
CRLF PROC NEAR
    PUSH DX
    PUSH AX
    MOV DL,0DH
    MOV AH,02H
    CALL BDOS      ;DO A <CR> AND <LF>
    MOV DL,0AH
    MOV AH,02H     ;SAVING MOST REGISTERS
    CALL BDOS
    POP AX
    POP DX
    RET
CRLF ENDP
CONV4  PROC NEAR
    MOV BX,OFFSET ADDR  ;AFTER IN4, CONVERTS 4 ASCII BYTES
    MOV CH,4       ;INTO A 2 BYTE ADDRESS (WORD)
CONV4A:  MOV AL,[BX]
    CALL ZCAH      ;USES AL,BX,CX
    AND AL,0FH
    MOV CL,4H
    SHL DX,CL
    OR DL,AL
    INC BX
    DEC CH
    JNZ CONV4A
    RET       ;RETURN WITH HEX ADDRESS IN DX
CONV2:   MOV BX,OFFSET ADDR
    MOV CH,2
CONV2A:  MOV AL,[BX]    ;AS ABOVE, ONLY AFTER IN2
    CALL ZCAH
    AND AL,0FH
    MOV CL,4H
    SHL DX,CL
    OR DL,AL
    INC BX
    DEC CH
    JNZ CONV2A
    RET
CONV4 ENDP
IN4   PROC NEAR
     MOV BX,OFFSET ADDR  ;GETS 4 ASCII BYTES INPUT FROM
    CALL ZIN       ;KEYBOARD, STORING SAME IN ADDR
    MOV [BX],AL
    INC BX
    CALL ZIN
    MOV [BX],AL
    INC BX
    CALL ZIN
    MOV [BX],AL
    INC BX
    CALL ZIN
    MOV [BX],AL
    CALL CRLF
    RET
IN2:     MOV BX,OFFSET ADDR  ;SEE ABOVE
    CALL ZIN
    MOV [BX],AL
    INC BX
    CALL ZIN
    MOV [BX],AL
     RET
IN4 ENDP
ZIN   PROC NEAR
    MOV AH,01H
    CALL BDOS
    RET
ZIN ENDP
BDOS PROC NEAR
PUSH SI
     PUSH ES
     PUSH DX
     PUSH CX
     PUSH BX
    INT 21H
    POP BX
    POP CX
    POP DX
    POP ES
POP SI
    RET
BDOS ENDP
MSG1 DB 'ENTER THE HEX WORD FOR THE OFFSET FROM EARLIER',13,10
DB 'PORTIONS OF THE PROGRAM (4 HEX DIGITS)',13,10,'$'
INMSG DB 'THIS PROGRAM COUNTS THE NUMBER OF RECORDS',13,10
DB ' OF 128 BYTES FOR USE IN RANDOM ACCESS FILES',13,10,'$'
FNFMSG DB 'FILE NOT FOUND, RETURNING TO DOS',13,10,'$'
ERR3MSG DB 'NO PARAMETERS ENTERED, RETURNING TO DOS',13,10
DB 'THE CORRECT FORMAT IS',13,10
DB 'COUNT FILE.NAM',13,10,13,10,'$'
```

```
SHIFT DW 00H
ADDR DB 8 DUP(0)
DB 0
CNTBUF DW  512 DUP(0)
BUFFER EQU $
START ENDP
CODE ENDS
END START
```

APPENDIX E

```
FCB EQU 005CH
DTA EQU 0080H
CODE SEGMENT PARA PUBLIC 'CODE'
ORG 0100H
START PROC FAR
ASSUME CS:CODE
ASSUME ES:CODE
ASSUME DS:CODE ;ESTABLISH OUR DATA SEGMENT
MOV BX,OFFSET DTA     ;CHECK HERE FOR PARAMETER
        MOV AL,[BX]
        OR AL,AL       ;IF ZERO,NO PARM ENTERED
        JNZ STEP1
        JMP INERR      ;IN WHICH CASE FLAG AN INPUT ERROR
STEP1:   MOV DX,OFFSET DESCMSG
MOV AH,9
CALL BDOS
CALL CRLF
MOV AH,1
CALL BDOS
CALL CRLF
MOV DX,OFFSET FCB
    MOV AH,17      ;SEARCH FOR FIRST
    CALL BDOS      ;SEE IF THE FILE EXISTS
    INC AL
    JNZ STEP2      ;0=FILE NOT FOUND
    JMP FNFERR          ;REPORT IF NOT FOUND
STEP2:    MOV DI,OFFSET FCB2+1
    MOV SI,OFFSET FCB+1
    MOV CX,8
    REP MOVSB      ;COPY FILE NAME FROM INPUT PARM
    MOV DI,OFFSET FCB2+9
    MOV AL,'A'
    MOV [DI],AL
     INC DI
    MOV [DI],AL
      INC DI
    MOV [DI],AL         ;MAKE FILE TYPE AAA=ASCII
STEP3:   MOV DX,OFFSET FCB
    MOV AH,15      ;OPEN FILE
    CALL BDOS      ;OPEN INPUT FILE
    MOV DX,OFFSET FCB2
    MOV AH,17      ;SEARCH FOR FIRST
    CALL BDOS      ;SEE IF SUCH A FILE ALREADY EXISTS
    INC AL
    JZ STEP31
    MOV DX,OFFSET FCB2
    MOV AH,19
    CALL BDOS      ;DELETE OLD FILE
STEP31:  MOV DX,OFFSET FCB2
    MOV AH,22      ;MAKE FILE
    CALL BDOS      ;CREATE THE ASCII FILE
    MOV BX,OFFSET BUFFER1
    MOV WORD PTR LOCALB,BX   ;SAVE OFFSET INTO BUFFER IN LOCALB
STEP4:   MOV DX,OFFSET FCB
    MOV AH,20
    CALL BDOS      ;READ SEQUENTIAL
    CMP AL,1
    JNZ STEP41
    JMP STEP5      ;END OF FILE
STEP41:  MOV BX,OFFSET LOCALB
    MOV BX,[BX]         ;GET BUFFER OFFSET IN BX
    MOV DH,40H          ;DMABUFER LENGTH IN WORDS
    MOV DL,08H      ;COUNTER FOR <CR>,<LF>
    MOV SI,OFFSET DTA
STEP42: MOV AL,'D'
MOV [BX],AL
INC BX
MOV AL,'W'
MOV [BX],AL
INC BX
MOV AL,' '
MOV [BX],AL
INC BX
STEP421: MOV AX,[SI]         ;TRANSFER HEX BYTE FROM DMAADD
OR AX,AX
JNZ STEP422
JMP STEP50
STEP422: CALL WORDASC      ;CONVERT IT TO  ASCII BYTES
    INC SI
    INC SI
    DEC DL
    JNZ STEP43          ;UNTIL END OF DMABUFFER
    DEC BX ;DELETE THE LAST COMMA
MOV AX,0A0DH
    MOV [BX],AX         ;CR+LF
    INC BX
     INC BX        ;ADD A CR,LF EVERY 8 WORDS
    MOV DL,8      ;RESET CRLF COUNTER
    DEC DH
JZ STEP431
JMP STEP42
STEP43:  DEC DH
    JMP STEP421
STEP431: MOV WORD PTR LOCALB,BX
    JMP STEP4      ;GET NEXT SECTOR AND REPEAT
STEP50: DEC BX
MOV AX,0A0DH
MOV [BX],AX
INC BX
INC BX
MOV WORD PTR LOCALB,BX
STEP5:   MOV BX,OFFSET LOCALB
    MOV BX,[BX]
    MOV AL,1AH          ;AT END OF FILE FILL WITH ^Z
    MOV CL,0FFH
STEP51:  MOV [BX],AL
    INC BX
    DEC CL
    JNZ STEP51
    MOV BX,OFFSET BUFFER
    MOV WORD PTR LOCALB,BX
    MOV DX,BX      ;GET NEW DMAADD FROM BUFFER
STEP52:  MOV AH,26
    CALL BDOS
    MOV DX,OFFSET FCB2  ;AND WRITE SEQUENTIAL
    MOV AH,21
    CALL BDOS
    MOV BX,LOCALB
    ADD BX,80H          ;ADVANCING THE DMAADD AS WE GO
    MOV WORD PTR LOCALB,BX
     MOV DX,BX
    MOV AL,[BX]
    CMP AL,1AH          ;CHECK EACH NEW SECTOR FOR EOF MARK
    JZ STEP53
    JMP STEP52          ;AND WRITE UNTIL EOF IS ENCOUNTERED
STEP53:  MOV DX,OFFSET FCB2
    MOV AH,16
    CALL BDOS     ;THEN CLOSE THE FILE
    INT 20H             ;AND RETURN TO DOS, A FAR RETURN
WORDASC PROC NEAR
PUSH AX
MOV AL,AH
CALL ZCHA
MOV [BX],CH
INC BX
MOV [BX],CL
INC BX
POP AX
CALL ZCHA
MOV [BX],CH
INC BX
MOV [BX],CL
INC BX
MOV AL,'H'
MOV [BX],AL
INC BX
MOV AL,','
MOV [BX],AL
INC BX
RET
WORDASC ENDP
INERR:   CALL CRLF
    MOV DX,OFFSET STRTMSG
    MOV AH,9
    CALL BDOS     ;NO PARAMETER ENTERED ERROR
    INT 20H
BDOS PROC NEAR
    PUSH ES
    PUSH DX
PUSH CX
PUSH BX
INT 21H
     POP BX
POP CX
POP DX
POP ES              ;SAVE ES IN BDOS CALLS
    RET
BDOS ENDP
FNFERR:  CALL CRLF
    MOV DX,OFFSET FNFMSG
    MOV AH,9
    CALL BDOS        ;FILE NOT FOUND MESSAGE
    INT 20H
CRLF PROC NEAR
    MOV DL,0DH
    MOV AH,02H
    CALL BDOS
    MOV DL,0AH
    MOV AH,02H
    CALL BDOS        ;ENTER <CR> AND <LF>
    RET
CRLF ENDP
ZCHA PROC NEAR
    PUSH AX
    AND AL,0F0H
    MOV CL,4
    ROL AL,CL        ;EXCHANGE LEFT AND RIGHT NYBBLES
    CALL CONVERT          ;CONVERT HEX DIGIT TO ASCII BYTE
    MOV CH,AL
    POP AX
    AND AL,0FH
    CALL CONVERT          ;CONVERT BOTH NYBBLES
    MOV CL,AL
    RET
CONVERT: ADD AL,90H
    DAA
    ADC AL,40H
    DAA              ;OPTIMIZED HEX DIGIT TO ASCII BYTE
    RET
ZCHA ENDP
DESCMSG DB 'CONVERTS A HEX FILE TO EQUIVALENT 7 BIT ASCII',13,10
DB 'FORMAT FOR TRANSMISSION TO THE MAINFRAME OR OTHER',13,10
DB 'DESTINATION RESTRICTED TO 7 BIT ASCII',13,10,13,10
DB 'THE FILE IS SAVED AS .AAA WITH SAME FILENAME',13,10
DB 'ENTER ANYTHING TO CONTINUE',13,10,'$'
STRTMSG  DB 'THE CORRECT FORMAT IS',13,10
         DB 'HEXASC FILE.NAM',13,10
         DB 'START OVER$'
FNFMSG   DB 'FILE NOT FOUND; RETURNING TO DOS$'
DIRMSG   DB 'NO DIRECTORY SPACE LEFT$'
DSKMSG   DB 'NO DISK SPACE LEFT$'
DMAADD   DB  80H  DUP(0)     ;DEFAULT DMAADD
LOCALB   DW   0000H          ;RESERVE SPACE FOR LOCAL STORAGE
FCB2     DB   36  DUP(0)        ;FCB FOR .AAA FILE
BUFFER   DB 'DW 0000',13,10
BUFFER1 EQU $               ;RESERVE SPACE FOR THE .AAA FILE
START ENDP
CODE ENDS
END START
```

APPENDIX F

```
;fixed the zeroing of the current record  number
;can now use repeatedly without rebooting; although it is
;recommended that a warm boot be done at the end of the session.
CODE SEGMENT PARA PUBLIC 'CODE'
ORG 0100H
ASSUME CS:CODE
ASSUME DS:CODE
ASSUME ES:CODE
start: jmp begin
filnam db 'vidram.dta',13
please db 'Please reboot at this stage',13,10
db 'enter any key to continue',13,10,'$'
begin: MOV AH,25H ;SET INTERRUPT VECTOR
MOV AL,05H
MOV DX,OFFSET RESIDE
CALL BDOS ;SET THE INTERRUPT 05H
MOV AH,31H ;TERMINATE BUT STAY RESIDENT
MOV AL,0 ;EXIT CODE
MOV CX,0000H ;GET THE WHOLE FILE
MOV BX,OFFSET LAST
SUB BX,CX
MOV CL,4
SHR BX,CL
INC BX ;ROUND UP TO THE NEXT INTEGER
MOV DX,BX ;DX=MEM SIZE IN PARAGRAPHS
CALL BDOS ;TERMINATE HERE
RESIDE: push ds
push es
push ax
push bx
push cx
push dx
push si
push di
MOV AX,CS ;THE INTERRUPT CHANGES ONLY THE CS REGISTER
MOV DS,AX ;IN ORDER TO ACCOMODATE A DS: FETCH
MOV ES,AX ;WE MUST BE SURE TO HAVE DS=CS, AND
mov dx,0080h
mov ah,1ah
```

```
call bdos ;set this dta
mov bx,005ch ;fcb
mov al,0
mov 32[bx],al ;zero out the current record number
mov si,offset filnam
mov di,5ch ;fcb
mov al,0fh
mov ah,29h ;parse filename
call bdos
mov ah,11h
mov dx,005ch
call bdos ;search for first
or al,al
jnz reside1
jmp killit
reside1: mov ah,16h ;create the file
mov dx,005ch
call bdos
mov ah,0fh ;video interrupt, status call
int 10h
mov di,0080h
mov [di],ax ;cols/mode
inc di
inc di
mov [di],bx ;bh = display page
mov dx,005ch
mov ah,15h
call bdos ;write the header record
mov ax,0b800h
mov es,ax ;point to video ram
mov di,4080h ;total count
mov bx,0000h
reside2: mov cx,40h
        mov si,0080h ;dta
res3: mov ax,es:[bx]
mov [si],ax
inc si
inc si
inc bx
inc bx
dec cx
jnz res3
```

```
sub di,80h
jnz next
jmp done
next: mov dx,005ch
mov ah,15h ;writeseq
call bdos
jmp reside2
done: mov dx,005ch
mov ah,10h ;close the file
call bdos
mov dx,offset please
mov ah,09h
int 21h
mov ah,1
int 21h
pop di
pop si
pop dx
pop cx
pop bx
pop ax
pop es
pop ds
pop ax
pop ax
pop ax ;restore the stack
mov ah,4ch
mov al,0
int 21h
killit: mov ah,13h ;delete file
mov dx,005ch
call bdos
jmp reside1
BDOS PROC NEAR
     PUSH ES
     PUSH DX
     PUSH CX
     PUSH BX
    INT 21H
    POP BX
    POP CX
    POP DX
    POP ES
    RET
BDOS ENDP
LAST DB 00H
CODE ENDS
END start
```

APPENDIX G

```
code segment para public 'code'
org 0100h
start proc far
assume cs: code, es: code,ds: code
jmp begin
header db 80h
pakend dw 0
mode db 04h
pageno db 01h
nblanks db 00h
ndata db 00h
datwrd dw 0000h
begin: mov bx,80h ;dta
mov al,[bx]
or al,al
jnz start1 ;0 ==> no parms entered
jmp inerr
start1: mov ah,11h
mov dx,005ch ;the file in the fcb
int 21h ;search for first
or al,al
jz start2 ;0 ==> file found
jmp fnferr
start2: mov ah,0fh
mov dx,005ch
int 21h ;open the file
mov dx,offset buffer
rdlup: mov ah,1ah
int 21h ;set up buffer as DTA
mov ah,14h
push dx
mov dx,005ch
int 21h ;read a sector to the DTA
cmp al,01h
jz done
pop dx
add dx,80h
jmp rdlup
done: pop ax ;readjust the stack
```

```
mov ah,10h ;close this file here
mov dx,005ch
int 21h
mov bx,offset buffer
add bx,80h
mov di,offset packbuf
mov cx,4000h
mov dx,0
paklup: mov al,[bx]
cmp al,0ffh
jz pak1
mov [di],dx
inc di
inc di
mov dx,0 ;reset count
mov word ptr datwrd,di
inc di
inc di
pak4: mov al,[bx]
cmp al,0ffh
jz pak3
mov [di],al
inc dx
inc bx
inc di
dec cx
jnz pak4
jmp last
mov [di],al
pak1: inc dx
inc bx
dec cx
jnz paklup
jmp last
pak3: mov si,word ptr datwrd
mov [si],dx ;data count here
mov dx,0
jmp paklup
last: mov word ptr pakend,di
mov cx,offset packbuf
sub di,cx ;di ==> byte number in pakbuf
mov cx,7
```

```
shr di,cl ;divide by 128
mov cx,di
inc cx ;bump to account for fractions
mov di,0068h ;zero out the rest of the fcb
mov dl,24
mov al,0
last1: mov [di],al
inc di
dec dl
jnz last1
mov bx,005ch ;dial up the fcb
mov al,'P'
mov 1[bx],al
mov al,'A'
mov 2[bx],al
mov al,'K'
mov 3[bx],al
mov dx,005ch ;fcb
mov ah,16h ;create new file
int 21h
mov dx,offset packbuf
wlup: mov ah,1ah ;set dta
int 21h
push dx
mov dx,005ch
mov ah,15h ;write seq
int 21h
pop dx
add dx,80h
dec cx
jnz wlup
mov ah,10h ;close file
mov dx,005ch
int 21h
int 20h
fnferr: mov dx,offset fnfmsg
mov ah,9
int 21h
call crlf
int 20h
inerr: mov dx,offset errmsg
mov ah,9
```

```
int 21h ;report the lapse
call crlf
int 20h ;exit
crlf proc near
mov ah,1
mov dl,0dh
int 21h
mov dl,0ah
mov ah,1
int 21h
ret
crlf endp
fnfmsg db 'file not found, exiting to DOS',13,10,'$'
errmsg db 'no parameters entered, the correct format is',13,10
db 'DISPLAY FILE.NAM',13,10
db 'returning to DOS',13,10,'$'
buffer db 4080h dup(0)
packbuf db 4000h dup(0)
start endp
code ends
end start
```

APPENDIX H

```
STACK SEGMENT PARA STACK 'STACK'
DB 256 DUP(0)
STACK ENDS
DATA SEGMENT PARA PUBLIC 'DATA'
OPENF DB 0FH ;OPEN FILE REFERENCED IN THE FCB
CLOSEF DB 10H ;CLOSE FILE
SRCHFRST DB 11H ;SEARCH FOR FIRST OCCURRENCE
SRCHNEXT DB 12H ;SEARCH FOR NEXT OCCURRENCE
DELETEF DB 13H ;DELETE FILE
READS DB 14H ;READ SEQUENTIALLY
READR DB 21H ;READ RANDOM
WRITES DB 15H ;WRITE SEQUENTIALLY
MAKEF DB 16H ;MAKE FILE
SETDMA DB 1AH ;SET DISK TRANSFER ADDRESS
PARSE DB 29H ;PARSE FILENAME,  SEE PG 5-71 TECH.REF
SELDSK DB 0EH ;SELECT DRIVE
MOORE   DB 10,13,'       ENTER ANY KEY TO DISPLAY MORE$'
;**********************************************************************
INMSG DB 'PRESENTATION QUESTIONS IN SELECTED ORDER',13,10,13,10
;
;
;       change INMSG as appropriate
;
;**********************************************************************
db 00h
FNFMSG DB 'FILE NOT FOUND, RETURNING TO DOS',13,10,'$'
ERR3MSG DB 'NO PARAMETERS ENTERED, RETURNING TO DOS',13,10
DB 'THE CORRECT FORMAT IS',13,10
DB 'RNDM FILE.NAM',13,10,13,10,'$'
QUERY DB 'ENTER A NUMBER AS THREE DECIMAL DIGITS',13,10
;***********************************************************************
;Replace the 135 by one less then the LAST question number
DB '(000 THROUGH 135;THE NUMBERS DO NOT ALWAYS',13,10
;***********************************************************************
DB 'CORRESPOND TO THE QUESTION NUMBERS)',13,10
DB 'ONCE THE DESIRED NUMBER IS ENTERED, HIT ENTER.',13,10
DB 'TO DISPLAY ANSWERS, HIT ANY KEY; THEN ANY',13,10
DB 'KEY TO CONTINUE',13,10,'$'
ENTRY DB 'ENTER DESIRED NUMBER (CONTC = ^C TO TERMINATE)',13,10,'$'
```

```
ERRMSG DB 'NUMBER OUT OF RANGE, TRY AGAIN',13,10,'$'
DTA DB 80H DUP(0) ;PROGRAM'S DATA TRANSFER ADDRESS
FCB DB 37 DUP(0) ;PROGRAM'S FILE CONTROL BLOCK
FCB1 DB  37  DUP(0)
;********************************************************************
;in place of the MC2.ALI file insert your FILE.ALI
FILEX DB 'MC2.ALI',13
;********************************************************************
CSAVE DW 0
COLORON DB 1BH,'[1;33m$'     ;SET COLOR TO YELLOW
COLOROFF DB 1BH,'[00;00m$'    ;RESET MONITOR
COLORANS DB 1BH,'[1;31m$'     ;SET COLOR TO RED
BLANK db 1bh,'[2J$'      ;BLANK SCREEN
RNDSAV DW 00H
ADDR DB 4 DUP(0)
LUKUPTBL LABEL WORD
;********************************************************************
;in place of the MC2.AAA file, enter your own FILE.AAA
INCLUDE MC2.AAA
;********************************************************************
INBUFF DB 8
DB 00H
DB 8 DUP(0) ;FOR BUFFERED INPUT
;********************************************************************
;refer to step eight and place the number of the question which
;has a graph on line 66.  Place them in ascending order.  Leave
;255 at the end of the list.
PICDAT DB 255
;********************************************************************
PICNUM DB '000'
;********************************************************************
;change the 0 to the number of graphs in use (not counting 255)
PICCNT DB 0 ;THE NUMBER OF PICTURES
;********************************************************************
PROBNO DB 30H
BUFFER DB 4000H DUP(0)
DATA ENDS ;ALL OTHER DATA GOES IN HERE
CODE SEGMENT PARA PUBLIC 'CODE'
START PROC FAR
ASSUME CS:CODE
PUSH DS
MOV AX,0
```

```
PUSH AX ;RETURN ADDRESS TO THE PSP ON THE STACK
MOV AX,DATA
MOV ES,AX
ASSUME ES:DATA ;ES POINTS TO OUR PROG. SEGMENT
MOV SI,80H ;PSP DTA
MOV DI,OFFSET DTA
MOV CX,80
REP MOVSB ;TRANSFER DTA AREA TO OUR SEGMENT
MOV SI,5CH ;PCP FCB
MOV DI,OFFSET FCB
MOV CX,37
REP MOVSB ;TRANSFER ANY FILE PARAMETERS TOO
MOV DS,AX
ASSUME DS:DATA ;NOW POINT DS TO OUR SEGMENT
MOV AH,SETDMA
MOV DX,OFFSET DTA
CALL BDOS ;OPEN DMAADD
CALL CRLF
PUSH DS
POP ES ;ES = DS HERE
MOV AH,PARSE
MOV SI,OFFSET FILEX
MOV DI,OFFSET FCB
MOV AL,0FH
CALL BDOS ;SET UP FCB
  MOV BX,OFFSET INMSG
START1: MOV AL,[BX]
CMP AL,00H
JZ START2
CALL DISPASC            ;DISPLAY OPENING MESSAGE
INC BX
JMP START1
START2: MOV DX,OFFSET FCB
MOV AH,SRCHFRST
CALL BDOS
OR AL,AL ;00 ==>MATCHING FILENAME FOUND
JZ BEG1
JMP FNFERR
BEG1: MOV DX,OFFSET FCB
MOV AH,OPENF
CALL BDOS ;OPEN THE FILE
MOV DX,OFFSET QUERY
MOV AH,9
CALL BDOS
CALL CRLF
START3: MOV DX,OFFSET ENTRY
MOV AH,9
CALL BDOS
CALL CRLF
MOV DX,OFFSET INBUFF ;PREPARE TO GET BUFFERED INPUT
MOV AH,0AH
CALL BDOS
MOV BX,OFFSET INBUFF
MOV AL,1[BX] ;CHECK FOR 3 DIGIT ENTRY
CMP AL,3
JZ START30
JMP INERR  ;IF BAD INPUT, DISPLAY ERROR MESSAGE
START30: MOV AL,2[BX] ;THE HUNDREDS DIGIT
SUB AL,30H ;CONVERT TO A DECIMAL DIGIT
MOV DH,AL ;DH SHOULD READ 0,HUNDREDS DIGIT
MOV AL,3[BX]
SUB AL,30H
CALL ZEN
MOV DL,AL
MOV AL,4[BX]
SUB AL,30H
OR DL,AL ;FOLD IN THE REST
CALL DECTOHEX ;ON RETURN, DX SHOULD HAVE THE HEX INTEGER
MOV BYTE PTR PROBNO,DL
;************************************************************************
;Replace the FFh by the hex number of questions(less one). This
;number is initially set at its maximum (255).
CMP DL,FFH
;************************************************************************
JBE START31
JMP INERR
START31: MOV DL,BYTE PTR PROBNO
MOV CL,0
MOV SI,OFFSET PICDAT
MOV DH,BYTE PTR PICCNT
START312: MOV AL,[SI]
CMP AL,DL
JNZ START311
MOV AL,CL
MOV DL,CL  ;ALSO SAVE CL IN DL FOR LATER USE
CALL ZEN
CALL ZCHA
MOV BX,OFFSET PICNUM
MOV 1[BX],AL
MOV AL,DL ;CL HAS BEEN USED IN ZEN
CALL ZCHA
MOV 2[BX],AL
CALL PICFIX
START311: INC CL
INC SI
DEC DH
JNZ START312
START313: MOV DL,BYTE PTR PROBNO
MOV DH,0
MOV SI,OFFSET LUKUPTBL
MOV BX,DX
ADD BX,BX ;MULTIPLY BY 2
MOV AX,[BX+SI]
MOV CX,[BX+SI+2]
SUB CX,AX ;NUMBER OF SECTORS TO DISPLAY
MOV BX,OFFSET FCB
MOV 33[BX],AX ;SET UP THE RANDOM FIELD
MOV AX,0
MOV 12[BX],AX ;THE CURRENT BLOCK
BEG2: MOV AH,READR
MOV DX,OFFSET FCB
CALL BDOS ;READ A SECTOR
BEG3: MOV WORD PTR CSAVE,CX
CALL DISPLAY
MOV CX,WORD PTR CSAVE
DEC CX
MOV WORD PTR CSAVE,CX
JNZ BEG31
MOV AH,08H
CALL BDOS
MOV DX,OFFSET BLANK
MOV AH,9
CALL BDOS
JMP START3
BEG31: MOV SI,OFFSET FCB
MOV AX,33[SI]
```

```
INC AX
MOV 33[SI],AX ;BUMP THE SECTOR COUNTER
JMP BEG2
PICFIX PROC NEAR ;NAMING THE VIDIO FILES,PAK.000 ETC.
        MOV BX,OFFSET FCB1
MOV AL,'P'
MOV 1[BX],AL
MOV AL,'A'
MOV 2[BX],AL
MOV AL,'K'
MOV 3[BX],AL
MOV CL,5
MOV AL,' '
ADD BX,4
PICFIX1: MOV [BX],AL ;INITIALIZE FILE CONTROL BLOCK
INC BX
DEC CL
JNZ PICFIX1
MOV SI,OFFSET PICNUM
MOV DI,BX
MOV CX,3
REP MOVSB
MOV DX,OFFSET FCB1
MOV AH,OPENF
CALL BDOS
RET ;THE PIC FILE IS OPEN AND READY TO SHOW
PICFIX ENDP
INERR: MOV DX,OFFSET ERRMSG ;ERROR MESSAGE IF INPUT WAS OUT OF RANGE
MOV AH,9
CALL BDOS
CALL CRLF
JMP START3
DECTOHEX PROC NEAR ;TURN DECIMAL NUMBER TO HEX NUMBER
MOV AH,0
MOV BX,0
PUSH DX
AND DX,000FH
MOV BL,DL ;CONSTRUCT THE HEX INTEGER IN BX
POP DX
PUSH DX ;PREPARE NEXT DIGIT
AND DX,00F0H ;THE TENS DIGIT
MOV CL,4
```

```
SHR DX,CL ;GET THE TENS DIGIT INTO DL
MOV AL,DL
MOV CH,0AH
MUL CH ;MULTIPLY BY 10
ADD BX,AX ;RUNNING TOTAL IN BX
POP DX
PUSH DX
AND DX,0F00H
MOV CL,8
SHR DX,CL
MOV AL,DL
MOV CH,100
MUL CH ;THE HUNDREDS PLACE
ADD BX,AX
POP DX
MOV DX,BX
RET
DECTOHEX ENDP
DISPLAY PROC NEAR ;DISPLAY QUESTION TO SCREEN
MOV DX,OFFSET COLORON ;SET COLOR TO YELLOW
MOV AH,9
CALL BDOS
MOV SI,OFFSET DTA ;START OF THE 128 BYTE DATA
MOV CL,80H
DISP2: MOV AL,[SI]    ;GET BYTE
CMP AL,02H ;CHECK FOR ^B, MORE
JNZ DISP22
CALL MORE
;********************************************************************
;If flag for graphs is to change, replace * in line 281 with new
;flag such as \.  See line 451 for a similar change.
DISP22: CMP AL, '*'
;********************************************************************
JNZ DISP21 ;IF NO GRAPH, CHECK ^C FOR END OF QUESTION
INC SI
MOV AH,8
INT 21H     ;WAIT HERE TO READ QUESTION
CALL PICDISP
DISP210: MOV AH,1
INT 21H ;GET KEYBOARD INPUT
CMP AL,'Q'-'@' ;JUMP BACK TO QUESTION
JZ QUES
```

```
CMP AL,'A'-'@' ;JUMP TO AWAIT ANSWER
JZ ANS
JMP DISP210 ;ACCEPT ONLY QUEST OR ANS
QUES:   MOV AH,0 ;RESTORE TEXT MODE
MOV AL,3
INT 10H ;RESTORE ALPHA MODE
POP AX ;PREPARE TO EXIT THE NEAR CALL
JMP START31 ;DISPLAY QUESTION AGAIN
ANS: MOV AH,0 ;RESTORE TEXT MODE TO DISPLAY ANSWER
MOV AL,3
INT 10H
DISP21: MOV AL,[SI]
CMP AL,03H ;CHECK FOR ^C FOR BEGINNING OF ANSWER
JZ DISPANS
CALL DISPASC ;PRINT SAME
INC SI
DEC CL
JNZ DISP2
MOV DX,OFFSET COLOROFF
MOV AH,9
CALL BDOS
RET ;DONE WITH THIS SECTOR
DISPANS: MOV AH,08H
CALL BDOS ;WAIT FOR ANY KEYPRESS
MOV DX,OFFSET COLOROFF
MOV AH,9
CALL BDOS
MOV DX,OFFSET COLORANS
MOV AH,9
CALL BDOS ;SET COLOR TO RED
INC SI
DEC CL ;MOVE PAST THE ETX
JNZ DISPA1
DISPANS1: MOV CX,WORD PTR CSAVE
DEC CX
MOV WORD PTR CSAVE,CX
MOV SI,OFFSET FCB
MOV AX,33[SI]
INC AX
MOV 33[SI],AX
MOV AH,READR
MOV DX,OFFSET FCB
```

```
CALL BDOS
MOV CL,80H
MOV SI,OFFSET DTA
DISPA1: MOV AL,[SI]
CMP AL,0AH ;LOOK FOR CARRIAGE RETURN IN ANSWER
JZ DISPA2
CALL DISPASC
INC SI
DEC CL
JNZ DISPA1
JMP DISPANS1
DISPA2: MOV AL,[SI]
CALL DISPASC
INC SI
DEC CL
JNZ DISPA2
MOV DX,OFFSET COLOROFF
MOV AH,9
CALL BDOS
RET
DISPLAY ENDP
MORE PROC NEAR ;PROCEDURE IF TEXT IS MORE THAN ONE SCREEN
PUSH DX
PUSH CX
PUSH SI
MOV DL,0DH
MOV AH,2
CALL BDOS
MOV AL,0AH
MOV AH,2
CALL BDOS
MOV DX,OFFSET MOORE
MOV AH,9
CALL BDOS ;PRINT THE "MORE" MESSAGE
MOV AH,1
CALL BDOS
POP SI
POP CX
POP DX
RET
MORE ENDP
DISPASC PROC NEAR ;DISPLAY TO SCREEN IN ASCII
```

```
CMP AL,0DH
JZ DASC2
CMP AL,0AH
JZ DASC2
CMP AL,09H
JZ DASC2
CMP AL,20H ;IGNORE ALL CONTROL CODES EXCEPT
JB DASC1 ;<CR>,<LF>, AND <HT>
DASC2: MOV DL,AL
MOV AH,2
CALL BDOS
RET
DASC1: MOV DL,20H
MOV AH,2
CALL BDOS
RET
DISPASC ENDP
PICDISP PROC NEAR ;PROCEDURE TO DISPLAY GRAPHS
PIC0: PUSH SI
PUSH CX
MOV BX,OFFSET FCB1
MOV AL,0
ADD BX,32
MOV [BX],AL ;RESET CURRENT RECORD FOR LOOPING PURPOSES
MOV AH,0
MOV AL,4
INT 10H ;SET UP MODE4
CALL BUFFNULL
MOV DX,OFFSET BUFFER
PICLUP: MOV AH,1AH ;SET DTA
INT 21H
MOV AH,14H ;READS
PUSH DX
MOV DX,OFFSET FCB1
INT 21H
CMP AL,1
JZ DONE
POP DX
ADD DX,80H
JMP PICLUP ;READIN THE PIC FILE TO BUFFER
DONE: POP AX ;RESET THE STACK
MOV SI,OFFSET BUFFER
MOV AX,0B800H
MOV ES,AX
MOV DI,0000
DONE1: MOV CX,[SI]
OR CX,CX
JZ LAST
INC SI
INC SI
MOV AL,0FFH
DONE2: MOV ES:[DI],AL
INC DI
DEC CX
JNZ DONE2
MOV CX,[SI]
MOV BP,DI
ADD BP,CX
CMP BP,4000H
JG LAST
INC SI
INC SI
REP MOVSB
JMP DONE1
LAST: PUSH DS
POP ES
POP CX
POP SI
MOV DX,OFFSET DTA
MOV AH,SETDMA
INT 21H ;RESET THE DTA
MOV AL,[SI]
;*****************************************************************
;If flag was changed in line 280, then make same change to line
;451(replace * with \ for example)
CMP AL,'*' ;ARE THERE MORE GRAPHICS PAGES?
;*****************************************************************
JZ NXTPAGE
JMP LASTLAST
NXTPAGE: MOV AH,8
INT 21H ;PAUSE BETWEEN PAGE CHANGES
INC SI
MOV BX,OFFSET FCB1
MOV AL,11[BX] ;NEXT PAGE
```

```
INC AL
MOV 11[BX],AL
MOV DX,BX
MOV AH,OPENF
CALL BDOS ;GET READY TO DISPLAY IT
JMP PICO
LASTLAST: RET
PICDISP ENDP
BUFFNULL PROC NEAR
MOV BX,OFFSET BUFFER
MOV CX,4000H
MOV AL,0
BNULL1: MOV [BX],AL
INC BX
DEC CX
JNZ BNULL1
RET
BUFFNULL ENDP
FNFERR: MOV DX,OFFSET FNFMSG
MOV AH,9
CALL BDOS ;REPORT FILE NOT FOUND
CALL CRLF
RET ;FAR RETURN
ERR3: MOV DX,OFFSET ERR3MSG
MOV AH,9
CALL BDOS
RET ;FAR RETURN
;**********************************************************************
;           ROUTINE UTILITIES (NOT NECESSARILY ALL USED)
;**********************************************************************
ZCHA  PROC NEAR
    AND AL,0FH     ;CONVERT HEX DIGIT IN AL TO ASCII BYTE IN AL
    ADD AL,90H
    DAA
    ADC AL,40H     ;DIGIT IS IN LOW NYBBLE
    DAA
    RET       ;OPTIMIZED SUBROUTINE
ZCHA ENDP
ZEN   PROC NEAR
     MOV CL,04H     ;EXCHANGE NYBBLES
    ROL AL,CL
    RET       ;USES CL,AL
```

```
ZEN ENDP
CRLF PROC NEAR
    PUSH DX
    PUSH AX
    MOV DL,0DH
    MOV AH,02H
    CALL BDOS      ;DO A <CR> AND <LF>
    MOV DL,0AH
    MOV AH,02H     ;SAVING MOST REGISTERS
    CALL BDOS
    POP AX
    POP DX
    RET
CRLF ENDP
BDOS PROC NEAR    ;FOR DOS 3.0 AND HIGHER, BDOS IS SUPERSEEDED BY
PUSH SI   ;INT 21H ALONE.  BDOS IS USED FOR COMPATIBILITY
     PUSH ES    ;PURPOSES WITH VERSIONS OF DOS PRIOR TO 3.0
     PUSH DX
     PUSH CX
     PUSH BX
    INT 21H
    POP BX
    POP CX
    POP DX
    POP ES
POP SI
    RET
BDOS ENDP
START ENDP
CODE ENDS
END START
```

APPENDIX I

```
STACK SEGMENT PARA STACK 'STACK'
DB 256 DUP(0)
STACK ENDS
DATA SEGMENT PARA PUBLIC 'DATA'
OPENF DB 0FH ;OPEN FILE REFERENCED IN THE FCB
CLOSEF DB 10H ;CLOSE FILE
SRCHFRST DB 11H ;SEARCH FOR FIRST OCCURRENCE
SRCHNEXT DB 12H ;SEARCH FOR NEXT OCCURRENCE
DELETEF DB 13H ;DELETE FILE
READS DB 14H ;READ SEQUENTIALLY
READR DB 21H ;READ RANDOM
WRITES DB 15H ;WRITE SEQUENTIALLY
MAKEF DB 16H ;MAKE FILE
SETDMA DB 1AH ;SET DISK TRANSFER ADDRESS
PARSE DB 29H ;PARSE FILENAME,  SEE PG 5-71 TECH.REF
SELDSK DB 0EH ;SELECT DRIVE
MOORE DB 10,13,'        ENTER ANY KEY TO DISPLAY MORE$'
;************************************************************************
INMSG DB 'PRESENTATION QUESTIONS IN RANDOM ORDER',13,10,13,10
;
;                 change INMSG as appropriate
;
;*************************************************************************
DB 00h
FNFMSG DB 'FILE NOT FOUND, RETURNING TO DOS',13,10,'$'
ERR3MSG DB 'NO PARAMETERS ENTERED, RETURNING TO DOS',13,10
DB 'THE CORRECT FORMAT IS',13,10
DB 'RNDM FILE.NAM',13,10,13,10,'$'
QUERY DB 'WHEN A GRAPH APPEARS, PRESS CONTROL Q TO TOGGLE BACK',13,10
DB 'TO QUESTION, CONTROL A TO TOGGLE BACK TO ANSWER',13,10,13,10
DB 'TO DISPLAY ANSWERS, HIT ANY KEY;',13,10
DB 'THEN ANY KEY TO CONTINUE',13,10,13,10
DB 'CONTROL C (^C) TERMINATES THE PROGRAM',13,10,13,10
DB 'PRESS ANY KEY TO CONTINUE',13,10,'$'
ENTRY DB 'ENTER DESIRED NUMBER (CONTC = ^C TO TERMINATE)',13,10,13,10
DB 'ENTER ANY KEY TO CONTINUE',13,10,'$'
ERRMSG DB 'NUMBER OUT OF RANGE, PLEASE REBOOT',13,10,'$'
DTA DB 80H DUP(0) ;PROGRAM'S DATA TRANSFER ADDRESS
FCB DB 37 DUP(0) ;PROGRAM'S FILE CONTROL BLOCK
```

```
FCB1 DB  37  DUP(0)
;********************************************************************
;
;
;
;in place of the mc2.ali file insert your file.ali
FILEX DB 'MC2.ALI',13
;********************************************************************
CSAVE DW 0
COLORON DB 1BH,'[1;33m$' ;CHANGE COLOR TO YELLOW
COLOROFF DB 1BH,'[00;00m$' ;RESET COLOR
COLORANS DB 1BH,'[1;31m$' ;CHANGE COLOR TO RED
BLANK db 1bh,'[2J$' ;BLANK SCREEN
RNDSAV DW 00H
ADDR DB 4 DUP(0)
LUKUPTBL LABEL WORD
;*********************************************************************
;in place of the MC2.AAA file insert your file.aaa
INCLUDE MC2.AAA
;*********************************************************************
INBUFF DB 8
DB 00H
DB 8 DUP(0) ;FOR BUFFERED INPUT
;*********************************************************************
;Refer to step eight.  Place the number of the question which
;has a graph on line 58.  Place them in ascending order.  Leave
;255 at the end of the list.
PICDAT DB 255
;*********************************************************************
PICNUM DB '000'
;**********************************************************************
;change the zero to the number of graphs in use (not counting 255)
PICCNT DB 0 ;THE NUMBER OF PICTURES
;**********************************************************************
PROBNO DB 30H
BUFFER DB 4000H DUP(0)
DATA ENDS ;ALL OTHER DATA GOES IN HERE
CODE SEGMENT PARA PUBLIC 'CODE'
START PROC FAR
ASSUME CS:CODE
PUSH DS
MOV AX,0
```

```
PUSH AX ;RETURN ADDRESS TO THE PSP ON THE STACK
MOV AX,DATA
MOV ES,AX
ASSUME ES:DATA ;ES POINTS TO OUR PROG. SEGMENT
MOV SI,80H ;PSP DTA
MOV DI,OFFSET DTA
MOV CX,80
REP MOVSB ;TRANSFER DTA AREA TO OUR SEGMENT
MOV SI,5CH ;PCP FCB
MOV DI,OFFSET FCB
MOV CX,37
REP MOVSB ;TRANSFER ANY FILE PARAMETERS TOO
MOV DS,AX
ASSUME DS:DATA ;NOW POINT DS TO OUR SEGMENT
MOV AH,SETDMA
MOV DX,OFFSET DTA
CALL BDOS ;OPEN DMAADD
CALL CRLF
PUSH DS
POP ES ;ES = DS HERE
MOV AH,PARSE
MOV SI,OFFSET FILEX
MOV DI,OFFSET FCB
MOV AL,0FH
CALL BDOS ;SET UP FCB
  MOV BX,OFFSET INMSG
START1: MOV AL,[BX]
CMP AL,00H
JZ START2
CALL DISPASC
INC BX
JMP START1
START2: MOV DX,OFFSET FCB
MOV AH,SRCHFRST
CALL BDOS
OR AL,AL ;00 ==>MATCHING FILENAME FOUND
JZ BEG1
JMP FNFERR
BEG1: MOV DX,OFFSET FCB
MOV AH,OPENF
CALL BDOS ;OPEN THE FILE
MOV DX,OFFSET QUERY
MOV AH,9
CALL BDOS
MOV AH, 1
CALL BDOS
CALL CRLF
START3: CALL GETSEED ;GET RANDOM INTEGER
MOV BYTE PTR PROBNO,DL
;*********************************************************************
;Replace the FFh by the hex number of questions(less one).  This
;number is initially set at its maximum (255).
CMP DL,FFH
;*********************************************************************
JBE START31
JMP INERR
START31: MOV CL,0
MOV SI,OFFSET PICDAT
MOV DH,BYTE PTR PICCNT
START312: MOV AL,[SI]
CMP AL,DL
JNZ START311
MOV AL,CL
MOV DL,CL ;ALSO SAVE CL IN DL FOR LATER USE
CALL ZEN
CALL ZCHA
MOV BX,OFFSET PICNUM
MOV 1[BX],AL
MOV AL,DL ;CL HAS BEEN USED IN ZEN
CALL ZCHA
MOV 2[BX],AL
CALL PICFIX
START311: INC CL
INC SI
DEC DH
JNZ START312
START313: MOV DL,BYTE PTR PROBNO
MOV DH,0
MOV SI,OFFSET LUKUPTBL
MOV BX,DX
ADD BX,BX ;MULTIPLY BY 2
MOV AX,[BX+SI]
MOV CX,[BX+SI+2]
SUB CX,AX ;NUMBER OF SECTORS TO DISPLAY
```

```
MOV BX,OFFSET FCB
MOV 33[BX],AX ;SET UP THE RANDOM FIELD
MOV AX,0
MOV 12[BX],AX ;THE CURRENT BLOCK
BEG2: MOV AH,READR
MOV DX,OFFSET FCB
CALL BDOS ;READ A SECTOR
BEG3: MOV WORD PTR CSAVE,CX
CALL DISPLAY
MOV CX,WORD PTR CSAVE
DEC CX
MOV WORD PTR CSAVE,CX
JNZ BEG31
MOV AH,08H
CALL BDOS
MOV DX,OFFSET BLANK
MOV AH,9
CALL BDOS
JMP START3
BEG31: MOV SI,OFFSET FCB
MOV AX,33[SI]
INC AX
MOV 33[SI],AX ;BUMP THE SECTOR COUNTER
JMP BEG2
GETSEED PROC NEAR ;RANDOM INTEGER GENERATOR
MOV DX,40H
IN AL,DX
;**********************************************************************
;in 2 places, change FFh to the hex number of questions
;this is the random number generator
GET1: CMP AL,FFH
JBE EXIT1
SUB AL,FFH
;**********************************************************************
JMP GET1
EXIT1: MOV DL,AL
RET
GETSEED ENDP
PICFIX PROC NEAR ;DISPLAY GRAPHS
        MOV BX,OFFSET FCB1
MOV AL,'P' ;IF USING DIFFERENT GRAPH NAMES
MOV 1[BX],AL ;CHANGE THE P IN LINE 206
MOV AL,'A'
MOV 2[BX],AL
MOV AL,'K'
MOV 3[BX],AL
MOV CL,5
MOV AL,' '
ADD BX,4
PICFIX1: MOV [BX],AL
INC BX
DEC CL
JNZ PICFIX1
MOV SI,OFFSET PICNUM
MOV DI,BX
MOV CX,3
REP MOVSB
MOV DX,OFFSET FCB1
MOV AH,OPENF
CALL BDOS
RET ;THE PIC FILE IS OPEN AND READY TO SHOW
PICFIX ENDP
INERR: MOV DX,OFFSET ERRMSG
MOV AH,9
CALL BDOS
CALL CRLF
JMP START3
DECTOHEX PROC NEAR ;DECIMAL TO HEX CONVERSION
MOV AH,0
MOV BX,0
PUSH DX
AND DX,000FH
MOV BL,DL ;CONSTRUCT THE HEX INTEGER IN BX
POP DX
PUSH DX ;PREPARE NEXT DIGIT
AND DX,00F0H ;THE TENS DIGIT
MOV CL,4
SHR DX,CL ;GET THE TENS DIGIT INTO DL
MOV AL,DL
MOV CH,0AH
MUL CH ;MULTIPLY BY 10
ADD BX,AX ;RUNNING TOTAL IN BX
POP DX
PUSH DX
```

```
AND DX,0F00H
MOV CL,8
SHR DX,CL
MOV AL,DL
MOV CH,100
MUL CH ;THE HUNDREDS PLACE
ADD BX,AX
POP DX
MOV DX,BX
RET
DECTOHEX ENDP
DISPLAY PROC NEAR ;DISPLAY TO SCREEN
MOV DX,OFFSET COLORON
MOV AH,9
CALL BDOS
MOV SI,OFFSET DTA ;START OF THE 128 BYTE DATA
MOV CL,80H
DISP2: MOV AL,[SI] ;GET BYTE
CMP AL,02H ;CHECK FOR ^B, MORE THAN ONE SCREEN
JNZ DISP22
CALL MORE
;******************************************************************
;If flag for graphs is to change, replace * in line 274 with new
;flag such as \.  See line 444 for similar change.
DISP22: CMP AL, '*'
;*******************************************************************
JNZ DISP21 ;DISPLAY QUESTION
INC SI
MOV AH,8
INT 21H ;WAIT HERE TO READ QUESTION
CALL PICDISP
DISP210: MOV AH,1
INT 21H ;GET KEYBOARD INPUT
CMP AL,'Q'-'@' ;CHECK FOR CONTROL Q
JZ QUES
CMP AL,'A'-'@' ;CHECK FOR CONTROL A
JZ ANS
JMP DISP210 ;ACCEPT ONLY QUEST OR ANS
QUES:   MOV AH,0
MOV AL,3
INT 10H ;RESTORE ALPHA MODE
POP AX ;PREPARE TO EXIT THE NEAR CALL
```

```
JMP START313
ANS: MOV AH,0 ;BLANK SCREEN
MOV AL,3
INT 10H
DISP21: MOV AL,[SI]
CMP AL,03H ;CHECK FOR CONTROL C
JZ DISPANS ;IF CONTROL C, PRINT ANSWER TO SCREEN
CALL DISPASC ;PRINT SAME
INC SI
DEC CL
JNZ DISP2
MOV DX,OFFSET COLOROFF
MOV AH,9
CALL BDOS
RET ;DONE WITH THIS SECTOR
DISPANS: MOV AH,08H
CALL BDOS ;WAIT FOR ANY KEYPRESS
MOV DX,OFFSET COLOROFF
MOV AH,9
CALL BDOS
MOV DX,OFFSET COLORANS ;SET COLOR TO RED
MOV AH,9
CALL BDOS
INC SI
DEC CL ;MOVE PAST THE ETX
JNZ DISPA1
DISPANS1: MOV CX,WORD PTR CSAVE
DEC CX
MOV WORD PTR CSAVE,CX
MOV SI,OFFSET FCB
MOV AX,33[SI]
INC AX
MOV 33[SI],AX
MOV AH,READR
MOV DX,OFFSET FCB
CALL BDOS
MOV CL,80H
MOV SI,OFFSET DTA
DISPA1: MOV AL,[SI]
CMP AL,0AH ;CHECK FOR CARRIAGE RETURN
JZ DISPA2
CALL DISPASC
```

```
INC SI
DEC CL
JNZ DISPA1
JMP DISPANS1
DISPA2: MOV AL,[SI]
CALL DISPASC
INC SI
DEC CL
JNZ DISPA2
MOV DX,OFFSET COLOROFF
MOV AH,9
CALL BDOS
RET
DISPLAY ENDP
MORE PROC NEAR ;USE IF MORE THAN ONE SCREENFUL
PUSH DX
PUSH CX
PUSH SI
MOV DL,0DH
MOV AH,2
INT 21H
MOV DL,0AH
MOV AH,2
INT 21H
MOV DX,OFFSET MOORE
MOV AH,9
INT 21H
MOV AH,1
INT 21H
POP SI
POP CX
POP DX
RET
MORE ENDP
DISPASC PROC NEAR ;DISPLAY IN ASCII
CMP AL,0DH ;CHECK FOR CONTROL CODES
JZ DASC2
CMP AL,0AH
JZ DASC2
CMP AL,09H
JZ DASC2
CMP AL,20H ;IGNORE ALL CONTROL CODES EXCEPT
```

```
JB DASC1 ;<CR>,<LF>, AND <HT>
DASC2: MOV DL,AL
MOV AH,2
CALL BDOS
RET
DASC1: MOV DL,20H
MOV AH,2
CALL BDOS
RET
DISPASC ENDP
PICDISP PROC NEAR ;SHOW GRAPHS
PICO: PUSH SI
PUSH CX
MOV BX,OFFSET FCB1
MOV AL,0
ADD BX,32
MOV [BX],AL ;RESET CURRENT RECORD FOR LOOPING PURPOSES
MOV AH,0
MOV AL,4
INT 10H ;SET UP MODE4
CALL BUFFNULL
MOV DX,OFFSET BUFFER
PICLUP: MOV AH,1AH ;SET DTA
INT 21H
MOV AH,14H ;READS
PUSH DX
MOV DX,OFFSET FCB1
INT 21H
CMP AL,1
JZ DONE
POP DX
ADD DX,80H
JMP PICLUP ;READIN THE PIC FILE TO BUFFER
DONE: POP AX ;RESET THE STACK
MOV SI,OFFSET BUFFER
MOV AX,0B800H
MOV ES,AX
MOV DI,0000
DONE1: MOV CX,[SI]
OR CX,CX
JZ LAST
INC SI
INC SI
MOV AL,0FFH
DONE2: MOV ES:[DI],AL
INC DI
DEC CX
JNZ DONE2
MOV CX,[SI]
MOV BP,DI
ADD BP,CX
CMP BP,4000H
JG LAST
INC SI
INC SI
REP MOVSB
JMP DONE1
LAST: PUSH DS
POP ES
POP CX
POP SI
MOV DX,OFFSET DTA
MOV AH,SETDMA
INT 21H ;RESET THE DTA
MOV AL,[SI]
;*******************************************************************
;If flag was changed in line 274, then make same changes to line
;444(replace * with \ for example).
CMP AL,'*' ;ARE THERE MORE GRAPHICS PAGES?
;*******************************************************************
JZ NXTPAGE
JMP LASTLAST
NXTPAGE: MOV AH,8
INT 21H ;PAUSE BETWEEN PAGE CHANGES
INC SI
MOV BX,OFFSET FCB1
MOV AL,11[BX] ;NEXT PAGE
INC AL
MOV 11[BX],AL
MOV DX,BX
MOV AH,OPENF
CALL BDOS ;GET READY TO DISPLAY IT
JMP PICO
LASTLAST: MOV BX,OFFSET FCB1
```

```
MOV AL,11[BX]
SUB AL,2
MOV 11[BX],AL ;RESET THE FCB1 FOR ANOTHER PASS THROUGH
MOV DX,BX
MOV AH,OPENF
INT 21H ;RE-OPEN THE MASTER FILE
RET
PICDISP ENDP
BUFFNULL PROC NEAR
MOV BX,OFFSET BUFFER
MOV CX,4000H
MOV AL,0
BNULL1: MOV [BX],AL
INC BX
DEC CX
JNZ BNULL1
RET
BUFFNULL ENDP
FNFERR: MOV DX,OFFSET FNFMSG ;FILE NOT FOUND ERROR MESSAGE
MOV AH,9
CALL BDOS ;REPORT FILE NOT FOUND
CALL CRLF
RET ;FAR RETURN
ERR3: MOV DX,OFFSET ERR3MSG   ;ERROR MESSAGE
MOV AH,9
CALL BDOS
RET ;FAR RETURN
ZCHA  PROC NEAR ;STANDARD SUBROUTINES FOLLOW, NOT ALL USED
    AND AL,0FH      ;CONVERT HEX DIGIT IN AL TO ASCII BYTE IN AL
    ADD AL,90H
    DAA
    ADC AL,40H   ;DIGIT IS IN LOW NYBBLE
    DAA
    RET ;OPTIMIZED SUBROUTINE
ZCHA ENDP
ZEN   PROC NEAR
     MOV CL,04H     ;EXCHANGE NYBBLES
    ROL AL,CL
    RET       ;USES CL,AL
ZEN ENDP
CRLF PROC NEAR
    PUSH DX
    PUSH AX
    MOV DL,0DH
    MOV AH,02H
    CALL BDOS      ;DO A <CR> AND <LF>
    MOV DL,0AH
    MOV AH,02H     ;SAVING MOST REGISTERS
    CALL BDOS
    POP AX
    POP DX
    RET
CRLF ENDP
BDOS PROC NEAR     ;CAN BE REPLACED AS INT 21 FOR DOS 3.0 OR HIGHER
PUSH SI ;FOR COMPATIBILITY, BDOS IS USED FOR ALL
     PUSH ES  ;VERSIONS OF DOS
     PUSH DX
     PUSH CX
     PUSH BX
    INT 21H
    POP BX
    POP CX
    POP DX
    POP ES
POP SI
    RET
BDOS ENDP
START ENDP
CODE ENDS
END START
```

APPENDIX J

```
code segment
org 100h
assume cs:code
mov ah,0
mov al,3 ;this is the desired mode number
int 10h ;video interrupt
int 20h ;terminate this fragment correctly
code ends
end
```

APPENDIX K

```
code segment
org 100h
assume cs:code
mov ah,0
mov al,4 ;this is the desired mode number
int 10h ;video interrupt
int 20h ;terminate this fragment correctly
code ends
end
```

APPENDIX L

```
code segment para public 'code'
org 0100h
start proc far
assume cs: code, es: code,ds: code
jmp begin
mode db 04h
pageno db 01h
begin: mov bx,80h ;dta
mov al,[bx]
or al,al
jnz start1 ;0 ==> no parms entered
jmp inerr
start1: mov ah,11h
mov dx,005ch ;the file in the fcb
int 21h ;search for first
or al,al
jz start2 ;0 ==> file found
jmp fnferr
start2: mov ah,0fh
mov dx,005ch
int 21h ;open the file
mov dx,offset buffer
rdlup: mov ah,1ah
int 21h ;set up buffer as DTA
mov ah,14h
push dx
mov dx,005ch
int 21h ;read a sector to the DTA
cmp al,01h
jz done
pop dx
add dx,80h
jmp rdlup
done: pop ax ;readjust the stack
mov si,offset buffer
mov ax,0b800h
mov es,ax
mov di,0000h
done1: mov cx,[si]
```

```
or cx,cx
jz last
inc si
inc si
mov al,0ffh
done2: mov es:[di],al
inc di
dec cx
jnz done2
mov cx,[si]
inc si
inc si
rep movsb
jmp done1
push cs
pop ds
last: int 20h
fnferr: mov dx,offset fnfmsg
mov ah,9
int 21h
call crlf
int 20h
inerr: mov dx,offset errmsg
mov ah,9
int 21h ;report the lapse
call crlf
int 20h ;exit
crlf proc near
mov ah,1
mov dl,0dh
int 21h
mov dl,0ah
mov ah,1
int 21h
ret
crlf endp
fnfmsg db 'file not found, exiting to DOS',13,10,'$'
errmsg db 'no parameters entered, the correct format is',13,10
db 'DISPLAY FILE.NAM',13,10
db 'returning to DOS',13,10,'$'
buffer db 4000h dup(0)
start endp
```

```
code ends
end start
```

APPENDIX M

```
STACK SEGMENT PARA STACK 'STACK'
DB 256 DUP(0)
STACK ENDS
DATA SEGMENT PARA PUBLIC 'DATA'
VIDFILE DB 'VIDRAM.DTA',0
OPENF DB 0FH ;OPEN FILE REFERENCED IN THE FCB
CLOSEF DB 10H ;CLOSE FILE
SRCHFRST DB 11H ;SEARCH FOR FIRST OCCURRENCE
SRCHNEXT DB 12H ;SEARCH FOR NEXT OCCURRENCE
DELETEF DB 13H ;DELETE FILE
READS DB 14H ;READ SEQUENTIALLY
READR DB 21H ;READ RANDOM
WRITES DB 15H ;WRITE SEQUENTIALLY
MAKEF DB 16H ;MAKE FILE
SETDMA DB 1AH ;SET DISK TRANSFER ADDRESS
PARSE DB 29H ;PARSE FILENAME,  SEE PG 5-71 TECH.REF
SELDSK DB 0EH ;SELECT DRIVE
INMSG DB 'SET ANY COLOR TO ANY OTHER SPECIFIC COLOR',13,10
DB 'IN MODE 4 GRAPHICS: 00==>BACKGROUND',13,10
DB '                    01==>FIRST COLOR',13,10
DB '                    10==>SECOND COLOR',13,10
DB '                    11==>THIRD COLOR',13,10
DB 'OF THE PALETTE CURRENTLY IN USE',13,10
DB 'THUS, TO CHANGE FIRST COLOR TO THIRD COLOR',13,10
DB 'ENTER 0111 FOLLOWED BY A <CR>',13,10,'$'
FNFMSG DB 'FILE NOT FOUND, RETURNING TO DOS',13,10,'$'
ERR3MSG DB 'NO PARAMETERS ENTERED, RETURNING TO DOS',13,10
DB 'THE CORRECT FORMAT IS',13,10
DB 'SETCOLOR XY UV',13,10,13,10
DB 'WHERE XY IS THE ORIGINAL COLOR, AND UV IS THE NEW COLOR',13,10,'$'
DTA DB 80H DUP(0) ;PROGRAM'S DATA TRANSFER ADDRESS
FCB DB 37 DUP(0) ;PROGRAM'S FILE CONTROL BLOCK
ADDR DB 4 DUP(0)
OLDCOL DB 0
NEWCOL DB 0
MASK1 DB 0
MASK2 DB 0
MASK3 DB 0
MASK4 DB 0
```

```
MASK11 DB 0
MASK12 DB 0
MASK13 DB 0
MASK14 DB 0
INBUFF DB 5
DB 0
DB 5 DUP(30H)
BUFFER DB 4080H DUP(0)
DATA ENDS ;ALL OTHER DATA GOES IN HERE
CODE SEGMENT PARA PUBLIC 'CODE'
START PROC FAR
ASSUME CS:CODE
PUSH DS
MOV AX,0
PUSH AX ;RETURN ADDRESS TO THE PSP ON THE STACK
MOV AX,DATA
MOV ES,AX
ASSUME ES:DATA ;ES POINTS TO OUR PROG. SEGMENT
MOV SI,80H ;PSP DTA
MOV DI,OFFSET DTA
MOV CX,80
REP MOVSB ;TRANSFER DTA AREA TO OUR SEGMENT
MOV SI,5CH ;PCP FCB
MOV DI,OFFSET FCB
MOV CX,37
REP MOVSB ;TRANSFER ANY FILE PARAMETERS TOO
MOV DS,AX
ASSUME DS:DATA ;NOW POINT DS TO OUR SEGMENT
MOV AH,SETDMA
MOV DX,OFFSET DTA
CALL BDOS ;OPEN DMAADD
CALL CRLF
PUSH DS
POP ES ;ES = DS HERE
MOV DX,OFFSET INMSG
MOV AH,9
INT 21H ;DISPLAY INTRO
CALL CRLF
MOV AH,10
MOV DX,OFFSET INBUFF
INT 21H ;GET USER INPUT
START1: MOV BX,OFFSET INBUFF
```

```
MOV AL,1[BX] ;GET FIRST PARAMETER
OR AL,AL
JNZ START2
JMP ERR3
START2: MOV AL,2[BX]
SUB AL,30H ;CONVERT TO HEX DIGIT
SHL AL,1 ;MAKE IT HIGH BIT
MOV AH,AL
MOV AL,3[BX]
SUB AL,30H
ADD AL,AH ;FORM THE BYTE
MOV BYTE PTR OLDCOL,AL ;SAVE SAME
MOV AL,4[BX]
SUB AL,30H
SHL AL,1
MOV AH,AL
MOV AL,5[BX]
SUB AL,30H
ADD AL,AH
MOV BYTE PTR NEWCOL,AL ;SECOND PARAMETER IS THE NEW COLOR
MOV SI,OFFSET VIDFILE
MOV DI,OFFSET FCB
MOV AL,0FH
MOV AH,29H
INT 21H ;PARSE VIDFILE
MOV DX,OFFSET FCB
MOV AH,SRCHFRST
INT 21H
OR AL,AL ;0 ==>SUCCESS
JZ NEXT
JMP FNFERR
NEXT: MOV AH,OPENF
INT 21H ;ATTEMPT TO OPEN SAME
STRT2: MOV DX,OFFSET BUFFER
MOV CX,4080H
STRT3: MOV AH,SETDMA
INT 21H ;SET UP BUFFER TO RECEIVE VIDRAM.DTA
MOV AH,READS
PUSH DX
MOV DX,OFFSET FCB
INT 21H
CMP AL,1
```

```
JZ START21
POP DX
ADD DX,80H
  SUB CX,80H
JNZ STRT3
START21: MOV AL,BYTE PTR OLDCOL
MOV CL,6
SHL AL,CL
MOV BYTE PTR MASK1,AL
MOV AL,BYTE PTR OLDCOL
MOV CL,4
SHL AL,CL
MOV BYTE PTR MASK2,AL
MOV AL,BYTE PTR OLDCOL
MOV CL,2
SHL AL,CL
MOV BYTE PTR MASK3,AL
MOV AL,BYTE PTR OLDCOL
MOV BYTE PTR MASK4,AL
MOV AL,BYTE PTR NEWCOL
MOV CL,6
SHL AL,CL
MOV BYTE PTR MASK11,AL
MOV AL,BYTE PTR NEWCOL
MOV CL,4
SHL AL,CL
MOV BYTE PTR MASK12,AL
MOV AL,BYTE PTR NEWCOL
MOV CL,2
SHL AL,CL
MOV BYTE PTR MASK13,AL
MOV AL,BYTE PTR NEWCOL
MOV BYTE PTR MASK14,AL
BUFLOOP: MOV BX,OFFSET BUFFER + 80H
MOV CX,4000H
LUP: MOV DH,0
MOV AL,[BX]
MOV DL,AL ;SAVE A COPY IN DL
AND AL,0C0H ;ISOLATE ONE PIXEL
CMP AL,BYTE PTR MASK1
JZ CHCOL1
LOOP1:OR DH,AL ;BUILD THE NEW BYTE 2 BITS AT A TIME
```

```
MOV AL,DL ;GET ORIGINAL BYTE BACK
AND AL,30H ;SECOND PIXEL
CMP AL,BYTE PTR MASK2
JZ CHCOL2
LOOP2: OR DH,AL
MOV AL,DL
AND AL,0CH
CMP AL,BYTE PTR MASK3
JZ CHCOL3
LOOP3: OR DH,AL
MOV AL,DL
AND AL,03H
CMP AL,BYTE PTR MASK4
JZ CHCOL4
LOOP4: OR DH,AL
MOV [BX],DH ;REPLACE OLD COLORS WITH NEW IN WHOLE BYTE
INC BX
DEC CX
JNZ LUP
JMP SAVEIT
CHCOL1: MOV AL,BYTE PTR MASK11
JMP LOOP1
CHCOL2: MOV AL,BYTE PTR MASK12
JMP LOOP2
CHCOL3:MOV AL,BYTE PTR MASK13
JMP LOOP3
CHCOL4: MOV AL,BYTE PTR MASK14
JMP LOOP4
FNFERR: CALL CRLF
MOV DX,OFFSET FNFMSG
MOV AH,9
INT 21H
RET ;FAR RETURN TO DOS
ERR3: MOV DX,OFFSET ERR3MSG
MOV AH,9
CALL BDOS
RET ;FAR RETURN
SAVEIT: MOV BX,OFFSET FCB + 9
MOV AL,'N'
MOV [BX],AL
INC BX
MOV AL,'E'
MOV [BX],AL
INC BX
MOV AL,'W'
MOV [BX],AL ;SET UP NEW FILE NAME
MOV AH,MAKEF
MOV DX,OFFSET FCB
INT 21H ;CREATE SAME
MOV BX,OFFSET FCB
MOV AL,0
MOV 32[BX],AL ;RESET CURRENT RECORD
MOV DX,OFFSET BUFFER
MOV CX,4080H
WLUP: MOV AH,SETDMA
INT 21H ;OPEN BUFFER FOR TRANSFER
MOV AH,WRITES
PUSH DX
MOV DX,OFFSET FCB
INT 21H ;WRITE ONE SECTOR
POP DX
ADD DX,80H
SUB CX,80H
JNZ WLUP
MOV AH,CLOSEF
MOV DX,OFFSET FCB
INT 21H
RET
ZCHA  PROC NEAR
    AND AL,0FH     ;CONVERT HEX DIGIT IN AL TO ASCII BYTE IN AL
    ADD AL,90H
    DAA
    ADC AL,40H     ;DIGIT IS IN LOW NYBBLE
    DAA
    RET       ;OPTIMIZED SUBROUTINE
ZCHA ENDP
ZEN   PROC NEAR
     MOV CL,04H     ;EXCHANGE NYBBLES
    ROL AL,CL
    RET       ;USES CL,AL
ZEN ENDP
CRLF PROC NEAR
    PUSH DX
    PUSH AX
    MOV DL,0DH
    MOV AH,02H
    CALL BDOS      ;DO A <CR> AND <LF>
    MOV DL,0AH
    MOV AH,02H     ;SAVING MOST REGISTERS
    CALL BDOS
    POP AX
    POP DX
    RET
CRLF ENDP
BDOS PROC NEAR
PUSH SI
     PUSH ES
     PUSH DX
     PUSH CX
     PUSH BX
    INT 21H
    POP BX
    POP CX
    POP DX
    POP ES
POP SI
    RET
BDOS ENDP
START ENDP
CODE ENDS
END START
```

APPENDIX N

1. Which of the following describes an even function.

a: f(x) = f(-x)

b: f(x) = -f(x)

c: f(x) = -f(-x)

d: none of the above

[a]

2. Which of the following describes an odd function.

a: f(x) = f(-x)

b: f(x) = -f(x)

c: f(x) = -f(-x)

d: none of the above

[c]

3. The integral of an odd function over a symmetric interval is

a: π

b: ∞

c: 2 times the value of the integral from zero to the upper limit

d: 0

[d]

4. An even function is a reflection through which of the following.

a: x axis

b: y axis

c: origin

d: the line y = x

[b]

5. An odd function is a reflection through which of the following.

a: x axis

b: y axis

c: the origin

d: the line y = x

[c]

6. Which of the following graphs is an even function?

a: a

b: b

c: c

d: d

[a]

7. Which of the following graphs is an odd function?

a: a

b: b

c: c

d: d

\

[b]

8. The composite function f(g(x)) is the result of

a: f(x) * g(x)

b: f(x) acting on the values of g(x)

c: g(x) acting on the values of f(x)

d: f(x) + g(x)

[b, ref. page 31, Berkey]

10. Does f(g(x)) = g(f(x))?

a: yes

b: no

c: sometimes

[c, ref. page 31, Berkey]

11. The domain of the composite function f(g(x)) is the set of all x

a: in the domain of g

b: in the domain of f

c: in the domain of g for which the number u = g(x) lies in the domain of f

d: in the domain of f for which the number u = f(x) lies in the domain of g

[c, ref. page 31, Berkey]

12. The range of the composite function f(g(x)) is

a: the range of g

b: the range of f

c: contained in the range of g

d: contained in the range of f

[d]

13. What is the domain of the composite function f(g(x)) where f(x) = √x and g(x) = x + 4?

a: (- ∞ , + ∞)

b: {0, + ∞)

c: {-4, + ∞)

d: {-4, 4}

e: (0, + ∞)

[c, ref. page 31, Berkey]

14. What is the range of the composite function f(g(x)) where f(x) = sin(x) and g(x) = x^3?

a: {0, + ∞)

b: {-1, 1}

c: (- ∞ , + ∞)

d: {0, 1}

[b]

15. A tangent line

a: intersects a curve in at most one point

b: is the limiting position of a secant line c: is parallel to the x axis

d: is perpendicular to the x axis

[b, ref. page 46, Berkey]

16. The slope of a line tangent to a function at a point $(x,f(x))$ is

a: $\lim_{h\to 0} f(x + h)$

b: $\lim_{h\to 0}\{ f(x + h) - f(x)\}$

c: $\lim_{h\to 0} \{f(x + h) - f(x)\} / h$

d: $y / f(y)$

[c, ref. page 47, Berkey]

17. The slope of the tangent of $f(x) = x^2$ at the point (2,4) is

a: 0

b: 2

c: 4

d: 6

[c, ref. page 48, Berkey]

18. Which of the following is false?

a: $1 = \lim_{x\to a} f(x)$ implies that $f(x)$ is near 1 when x is near a

b: $\lim_{x\to a} f(x)$ exists implies that $f(a)$ exists

c: $\lim f(x)$ is determined by the behavior of f for x->a

x near a

d: $\lim_{x\to a} f(x)$ exists implies that $f(a+) = f(a-)$

[b, ref. page 51, Berkey]

19. Find $\lim_{x\to 2} \{x^2 - 3x + 2\} / \{x^2 + x - 6\}$.

a: 2

b: 4

c: .5

d: -3

e: .2

[e, ref. page 53, Berkey]

20. The formal definition of a limit is that the *number* l is the limit of the function f as x approaches a, written $l = \lim_{x\to a} f(x)$ if and only if, given any number $\epsilon > 0$ there exists a corresponding number $\delta > 0$ so that if $0 < |x - a| < \delta$, then $|f(x)-l| < \epsilon$.

a: true

b: false

[a, ref. page 57, Berkey]

21. Assume $\lim_{x\to a} f(x) = l$ and $\lim_{x\to a} g(x) = m$. Let c be any constant. Which of the following is false?

a: $\lim_{x\to a} \{f(x) + g(x)\} = l + m$

b: lim {c*f(x)} = c*l
x->a

c: lim {f(x)*g(x)} = l*m
x->a

d: lim {f(x)/g(x)} = m/l provided l <> 0
x->a

e: lim {f(x)}^n = l^n
x->a

[d, limit is l/m provided m <> 0, ref. page 62, Berkey]

22. Find lim (3x^4 + $7x^2$ + 4x).
x->2

a: 14

b: 34

c: 64

d: 84

[d, ref. page 63, Berkey]

23. Find lim (1/x^3 - 3/x^2 + 5x^3).
x->-2

a: -327/8

b: -8

c: -20

d: -60

[a, ref. page 63, Berkey]

24. Find lim (sin(x) / x).
x->0

a: 0

b: 1

c: + ∞

d: $-\infty$

[b, ref. page 66, Berkey]

25. Find $\lim_{x \to \pi/4} (\sin(x) / \tan(x))$.

a: $\pi/4$

b: $4/\pi$

c: $\sqrt{2}/2$

d: 1

e: 2

[c, sin(x) / tan(x) = cos(x)]

26. The function f is continuous at $x = a$ if f is defined on an open interval containing a and $f(a) = \lim_{x \to a} f(x)$.

a: true

b: false

[a, ref. page 76, Berkey]

27. Find the numbers x at which $f(x) = \{x^2 - 4\} / \{x - 2\}$ is continuous.

a: 2

b: 4

c: all x

d: all x <> 2

e: all x <> 4

[d, ref. page 77, Berkey]

28. For what values of x is $f(x) = \{x + 2\} / \{x^2 - x - 2\}$ discontinuous?

a: $x = -2$

b: $x = -1$

c: $x = -2$ and -1

d: $x = 2$ and -1

e: $x = 2$ and 1

[d]

29. For what values of x is $f(x) = \begin{matrix} -x & x <= -1 \\ 4 - x^2 & -1 < x <= 2 \\ \frac{1}{2}x - 1 & 2 < x \end{matrix}$ discontinuous?

a. $x = -1$

b. $x = 2$

c. $x = -1$ and 2

d. none of the above

[a]

30. The function $f(x) = x / \cos(x)$ is continuous on the open interval $(0, \pi)$.

a: true

b: false

[b]

31. The function $f(x) = |x|$ is continuous on the closed interval

{-3, 3}.

a: true

b: false

[a]

32. If the functions f and g are continuous at x = a and if c is any real number, then which of the following is false?

a: f + g is continuous

b: c*f is continuous

c: f*g is continuous

d: f/g is continuous provided g(a) <> 0

e: none of the above

[e, ref. page 78, Berkey]

33. The derivative of f(x) = 1 / {2x + 3} at x = 1 is

a: -2 / 3

b: -2 / 25

c: 1 / 5

d: 0

e: none of the above

[b]

34. The derivative of f(x) = |x| at x = 0 is

a: 0

b: 1

c: ½

d: not defined

[d]

35. The derivative of f(x) = cos(x) is

a: 1 / cos(x)

b: sin(x)

c: tan(x)

d: -sin(x)

[b]

36. The derivative of f(x) = 2x^3 + $6x^2$ - 5x - $\sqrt{x}$ is

a: $6x^2 + 12x - 5 - (.5 / \sqrt{x})$

b: 6x + 12x - 5x - ½x

c: $6x^2 + 12x + 5 + .5\sqrt{x}$

d: $6x^2 + 12x - 5 -.5\sqrt{x}$

[a]

37. The derivative of the function f on the interval I, denoted by f', is the function with values

$$f'(x) = \lim_{h \to 0} \{f(x + h) - f(x)\} / h$$

provided this limit exists for all x contained in I.

a: true

b: false

[a, ref. page 94, Berkey]

38. If the function s gives the position of an object moving

along a line, then which of the following describe the velocity of the object?

a: 1 / s

b: s}

c: the first derivative of s with respect to time

d: the second derivative of s with respect to time

[c, ref. page 113, Berkey]

39. How is speed related to velocity?

a: speed = 1 / velocity

b: speed = |velocity|

c: speed = velocity}

d: speed = first derivative of velocity

[b, ref. page 114, Berkey]

40. When the velocity of an object is zero, the position of the object is a constant.

a: true

b: false

[a]

41. Geometrically, the first derivative at a point a is a line which is parallel to the x axis and contains the point a.

a: true

b: false

[b]

42. Geometrically, the slope of a line tangent to a curve at a point a is the

a: first derivative

b: second derivative

c: asymptotic line

d: third derivative

[a]

43. What is the equation of the line tangent to the function $f(x) = x^2$ at the point (3,9)?

a: $y = 6x$

b: $y = 6x + 3$

c: $y = 6x - 9$

d: $y = 2x + 3$

[c]

44. What is the equation of the line tangent to the function $f(x) = \sin(x) + \cos(x) + 2$ at $x = \pi/2$?

a: $y = -x + 3 + \pi/2$

b: $y = -x$

c: $y = 2x + 3 + \pi/2$

d: $y = 2x - 3 + \pi/2$

[a]

45. What is the first derivative of $f(x) = \tan(x)$?

a: $\cos(x)$

b: -sin(x)

c: sec^2 (x)

d: cot(x)

e: -csc(x)

[c, ref. page 109, Berkey]

46. What is the first derivative of f(x) = {x^4 - 6x^2 }^3?

a: {4x^3 - 12x}^3

b: 3{x^4 - 6x^2 }^{2}

c: 12x^3 - 36x

d: 3*{x^4 - 6x^2 }*{4x^3 - 12x}

[d]

47. What is the first derivative of f(x) = sin(6x^2 - x)?

a: {12x - 1}*{cos(6x^2 -x)}

b: cos(6x^2 - x)

c: {12x - 1}*cos(x)

d: {12x - 1}*{sin(6x^2 - x}

[a, ref. page 123, Berkey]

48. If the function g is differentiable at x and the function f is differentiable at u = g(x), then the composite function {f composite g } is differentiable at x, and {f composite g}'(x) = g'(f(x))f'(x) {the chain rule}.

a: true

b: false

[b, {f composite g}'(x) = f'(g(x))g'(x), ref. page 122, Berkey]

49. What is the first derivative of f(x) = {x^3 - x^2 + 3}^ $\frac{1}{4}$?

a: {$\frac{1}{4}$} * {x^3 - x^2 + 3}^{-3/4}

b: {$3x^2$ - 2x}^{$\frac{1}{4}$}

c: {$\frac{1}{4}$} * {{x^3 -x^2 + 3}^{-3/4}} * {$3x^2$ - 2x}

d: none of the above

[c]

50. What is the slope of the line tangent to the graph of the ellipse {x^2 / 16} + {y^2 / 9} = 1 at the point {2, 3 { 3/2}?

a: 3 / 4

b: - {3 / 4

c: 2

d: -3 / 4

[b, ref. page 127, Berkey]

51. What is the slope of the line tangent to the graph of y^2 + x^2*y = $3x^2$ at the point {2, 2}?

a: 2

b: $\frac{1}{2}$

c: 4

d: 1

[d, ref. page 128, Berkey]

52. What is the minimum value of the function f(x) = x on the half open interval (-1, 1}?

a: 1

b: 0

c: -1

d: none of the above

[d, ref. page 150, Berkey]

53. A continuous function will always have both a maximum and a minimum value on a closed finite interval {a, b}.

a: true

b: false

[a, ref. page 150, Berkey]

54. What is the maximum value of $f(x) = 4 - x^2$ on the interval {-3, 3}?

a: -5

b: 0

c: 4

d: 5

[c]

55. Let f be a continuous function on the interval {a, b}, let f'(x) exist for each x in (a, b) and let f(a) = f(b). Does there exist at least one number c in (a, b) for which f'(c) = 0 ?

a: yes

b: no

[a, Rolle's Theorem, ref. page153, Berkey]

56. Let f be continuous on {a, b} and let f'(x) exist for each x in (a, b) and let there exist at least one number c in (a, b) for which f'(c) = {f(b) - f(a)} / {b - a}. This describes what theorem?

a: Rolle's theorem

b: mean-value theorem

c: intermediate value theorem

d: extreme value theorem

[b, ref. page 154, Berkey]

57. Let s(t) be a differentiable position function of an object. The average velocity from time t = a to time t = b equals the instantaneous velocity v(t) = s'(t) for at least one time t = c where c is between a and b. This an example of

a: Rolle's theorem

b: the mean-value theorem

c: the intermediate value theorem

d: the extreme value theorem

[b]

58. Let f(x) = x^(2/3) in the interval {-1, 1}. What part of the mean-value theorem is not satisfied for f?

a: f is not continuous for each x in {-1, 1}

b: f is not differentiable for each x in {-1,1}

c: f(2/3) does not exist in {-1,1}

d: {f(1) - f(-1)} / {1 - (-1)} does not exist

[b]

59. Let f(x) = {x on the interval {0, 4}. By the mean value theorem, a number c exists in the interval {0, 4} such that f'(c) = {f(4) - f(0)} / {4 - 0}. What is c?

a: ¼

b: ½

c: 1

d: 2

e: 4

[c]

60. Let f be defined over an interval I. Let x and y be elements of I. If x < y and f(x) > f(y) then

a: f is increasing from x to y

b: f is decreasing from x to y

c: f is constant from x to y

d: none of the above

[b, ref. page 158, Berkey]

61. Let f be continuous on the open interval I and let f' exist for all x in I. Then f'(x) < 0 for all x in I implies

a: f is increasing on I

b: f is decreasing on I

c: f is constant on I

d: none of the above

[b, ref. page 159, Berkey]

62. For a function f, those numbers c in the domain of f for which either f'(c) = 0 or f'(c) fails to exist are called

a: critical points

b: inflection points

c: extreme points

d: points of discontinuity

[a, ref. page 162, Berkey]

63. The second derivitive of a function describes

a: slope of tangent line

b: concavity of the function

c: critical points

d: none of the above

[b, ref. page 178, Berkey]

64. Let f(x) = (x + 3)^3. what are the inflection points of f?

a: 0

b: 3

c: -3

d: 6

[c]

65. Let f(x) = x^(2/3) - (1/5)x^(5/3). Over what interval is f concave downward?

a: (-1, + ∞)

b: (- ∞ , -1)

c: (-1, 0)

d: (0, + ∞)

[a, ref. page 181, Berkey]

66. The function f(x) = (x + 2) / x has an asymptote at the line

a: x = 1

b: y = 1

c: x = 0

d: y = -2

e: b and c

[e, ref. page 186, Berkey]

67. Use Newton's method to approximate the zero of the function f(x) = x^3 - 10 in the interval (0, 4).

a: 0

b: 2

c: 2.17

d: 2.68

e: 1.68

[c, ref.page 143, Berkey]

68. To approximate solutions to f(x) = 0,

Newton's method uses the approximation

(see graph)

a: true

b: false

\

[a, ref. page 143, Berkey]

69. Depending on the function and the initial approximation, Newton's method for solving for zeros of functions always converges to the desired zero?

a: true

b: false

[b, ref. page 145, Berkey]

70. Approximate the $\sqrt{37}$ by using a linear approximation to $f(x) = \sqrt{x}$ where x = 36.

a: 6

b: 6.08

c: 6.16

d: 6.32

[b]

71. The symbols dy and dx are referred to as

a: derivative

b: differentials

c: approximations

d: none of the above

[b, ref. page 141, Berkey]

72. dy = cos(x)dx is the differential form of

a: y = sin(x)

b: y = -sin(x)

c: y = cos(x)

d: y = tan(x)

[a, ref. page 141, Berkey]

73. If f(x) = ln(x) then f'(x) =

a: 1 / x

b: x

c: $-1 / x^2$

d: 1

e: none of the above

[a]

74. Assume that the rate of growth of a population of fruit flies is proportional to the size of the population at each instant of time. If 100 fruit flies are present initially and 300 are present after 10 days, how many will be present after 15 days?

a: 400

b: 450

c: 500

d: 520

[d, ref. page 406, Berkey]

75. Simplify e^{ln(3)}.

a: 3

b: 1/3

c: 9

d: -1/3

[a]

76. Let x and y be any real numbers. Let r be a rational number. which of the following is false:

a: e^x * e^y = e^{x + y}

b: e^x / e^y = e^{x - y}

c: {e^x}^r = e^{xr}

d: none of the above

[d, ref. page 390, Berkey]

77. Let a, x and y be any real numbers. Which of the following is false:

a: a^x a^y = a^{x + y}

b: a^x / a^y = a^{x - y}

c: {a^x}^y = a^{x y}

d: none of the above

[d, ref. page 398, Berkey]

78. The function y = tan(x) is periodic.

a: true

b: false

[a, ref. page 427, Berkey]

79. What is the domain of the principal branch of the function $y = \tan(x)$?

a: $\{0, \pi\}$

b: $(0, \pi)$

c: $(-\pi/2, \pi/2)$

d: $\{-1, 1\}$

[c, ref. page 427, Berkey]

80. What is arctan(1)?

a: π

b: $\pi/2$

c: $\pi/4$

d: $\pi/8$

[c, ref. page 428, Berkey]

81. What is arctan(0)?

a: 0

b: π

c: $\pi/2$

d: $\pi/4$

[a, ref. page 428, Berkey]

82. What is the range of the principal branch of arcsin(x)?

a: $\{-1, 1\}$

b: $(-1, 1)$

c: $\{0, \pi\}$

d: $\{-\pi/2, \pi/2\}$

[d, ref. page 429, Berkey]

83. What is the range of the principal branch of arccos(x)?

a: $\{0, \pi\}$

b: $\{-\pi/2, \pi/2\}$

c: $\{-1, 1\}$

d: $(-\infty, +\infty)$

[a, ref. page 429, Berkey]

84. What is the derivative of arcsin(x)?

a: $1 / \sqrt{(1 - x^2)}$ abs(x) < 1

b: $1 / (1 + x^2)$ $-1 < x < +1$

c: $-1 / \sqrt{(1 - x^2)}$ abs(x) < 1

d: $-1 / (1 + x^2)$ $-1 < x < +1$

[a, ref. page 433, Berkey]

85. What is the derivative of arccos(x)?

a: $1 / \sqrt{(1 - x^2)}$ abs(x) < 1

b: $1 / (1 + x^2)$ $-1 < x < +1$

c: $-1 / \sqrt{(1 - x^2)}$ abs(x) < 1

d: $-1 / (1 + x^2)$ $-1 < x < 1$

[c, ref. page 434, Berkey]

86. If f(x) = arctan(3x) then f'(x) is

a: 1 / {1 + 3x}

b: 1 / {1 + $9x^2$ }

c: 3 / {1 + $9x^2$ }

d: 3 / {1 + 3x}

[c, ref. page 433, Berkey]

87. What is the derivative of tanh(x)?

a: {sech^2 (x) }

b: -{csch(x) }^{2}

c: -sech(x) * tanh(x)

d: -csch(x) * coth(x)

[a, ref. page 440, Berkey]

88. Sinh(x) is defined as which of the following?

a: ½ * {e^x - e^(-x) }

b: 2 / {e^x - e^(-x) }

c: ln{x + (√(x^2 + 1) }

d: {e^x + e^(-x) } / 2

[d, ref. page 438, Berkey]

89. l'Hopital's rules are used to evaluate which of the following?

a: integrals

b: derivatives

c: limits of the form 0/0

d: limits of the form ∞/∞

e: c and d

[e, ref. page 484, Berkey]

90. Determine $\lim_{x \to 0} \{\sin(x) / x\}$ using L'Hopital's Rule.

a: 0

b: 1

c: π

d: $\pi/2$

[b, ref. page 485, Berkey]

91. What is the limit as x goes to zero of

$\{x - \tan(x)\} / \{x - \sin(x)\}$?

a: -2

b: π

c: 0

d: 1

[a, ref. page 486, Berkey]

92. What is the limit as x goes to infinity of

{x^2 + 5} / {x + e^x}?

a: 5

b: 1

c: 0

d: $\frac{1}{2}$

[c, ref. page 487, Berkey]

93. What is the limit as x goes to one of ln(x) / {x - 1}?

a: 1

b: ∞

c: 0

d: none of the above

[a, use L'Hopital's Rule]

94. What is the indefinite integral of 2x^3 - 4x^2 + 5x -2?

a: 6x^2 - 8x + 5

b: ½x^4 - 4/3x^3 + 5/2x^2 - 2x

c: ½x^4 + 4/3x^3 + 5/2x^2 + 2x

d: ½x^4 - 4/3x^3 + 5/2x^2 - 2x + constant

[d]

95. What is the indefinite integral of e^x?

a: x*e^x + e^x + constant

b: e^x + constant

c: xe^x + constant

d: e^x

[b]

96. What is the indefinite integral of tan(x)?

a: cot(x) + constant

b: sec(x) + constant

c: -ln |cos(x)| + constant

d: -cot(x) + constant

[c]

97. Evaluate the integral of (4x + 6) from 1 to 2.

a: 20

b: 14

c: 12

d: 4

[c]

98. Evaluate the integral of cos(x) from 0 to $\pi/2$.

a: 1

b: 0

c: -1

d: $\pi/4$

[a]

99. Evaluate the integral of $|x - 2|$ from -1 to 5.

a: 0

b: 12

c: 9

d: 3

[c]

100. Let f(x) = sin(x). Let g(t) = the integral of f(x) from a to t. What is g'(t)?

a: 1

b: sin(t)

c: -cos(t)

d: 0

[b, Fundamental Theorem of Calculus, ref. page 282, Berkey]

101. To approximate the integral from a to b of f(x)dx, the following formula can be used:(see graph) this formula is the

a: trapezoidal rule

b: midpoint rule

c: half angle formula

d: Simpson's rule\

[a, ref. page 307, Berkey]

102. Approximate the integral of (1/x) from 1 to 4 using the trapezoidal rule with *n* = 6.

a: .8

b: 1.0

c: 1.2

d: 1.6

[c, ref. page 307, Berkey]

103. When using Simpson's rule to approximate definite integrals, n, the number of subdivisions of the interval (a, b), must be odd.

a: true

b: false

[b, n must be even, ref. page 308, Berkey]

104. Approximate the integral of (1/x) from 1 to 4 of using Simpson's rule with n = 6.

a: 1.39

b: 1.24

c: .83

d: 4.17

[a, ref. page 309, Berkey]

105. Find the volume of the cone obtained by revolving about the x-axis the region bounded above by the graph of $f(x) = x/3$ and below by the x-axis for $0 < x < 3$.

a: 1.5

b: 1

c: π

d: 3

[c, ref. page 318, Berkey]

REFERENCE

Berkey, D.D., *Calculus*, Second Edition, W. B. Saunders, 1988.

INITIAL DISTRIBUTION LIST

1. Commandant of the Marine Corps 1
Code TE 06
Headquarters, U.S. Marine Corps
Washington, D.C. 20380-0001

2. Defense Technical Information Center 2
Cameron Station
Alexandria, Virginia 22304-6145

3. Library
Code 0142 2
Naval Postgraduate School
Monterey, California 93943-5002

4. Professor G. E. Latta 1
Code MA LZ
Department of Mathematics
Naval Postgraduate School
Monterey, California 93943

5. Professor H. M. Fredricksen 1
Code MA FS
Department of Mathematics
Naval Postgraduate School
Monterey, California 93943

6. Captain Matthew Lampugnano 1
4903 N. Oconto
Harwood Heights, Illinois 60656