Advanced COBOL Overview

Assignments

Lectures

Quizzes and Tests

Grades

The objective of this course is to give the COBOL student a broader base knowledge of the COBOL language, more confidence in solving business problems with COBOL and an understanding of the environments in which a COBOL programmer functions.

This course is administered via distance learning, and requires the self discipline and motivation to study material, solve business problems, and be responsible for submitting assignments without the framework of the typical classroom.

Assignments

Assignments are posted to the Micro Focus website on a weekly basis. The assignment will be accessible to students by Tuesday at 12:00 p.m. Students should read the assignments and download any associated/required material that might be linked to the assignment page.

Completed assignments are due on the Tuesday of the following week, by 12:00 p.m., unless otherwise noted by the instructor. Completed assignments are to be e-mailed to the instructor in the format specified in the assignment.

Lectures

There are no 'lectures' per se, in a distance learning class. Rather, the instructor and students communicate via e-mail, open chat sessions and weekly conference calls and individual phone conversations. Students should refer to the Advanced COBOL index page for a schedule of available instructor time for chat sessions and conference calls.

Quizzes and Tests

There are two tests in the Advanced COBOL course, a midterm and a final. In addition, students should expect three additional quizzes during the semester. All testing is based on knowledge of reading materials and solving business problems with COBOL. The business problems will be in the form of programs to be written and in some cases will require written explanations.

Grades

Final grades are determined according to the following schedule:

35%	Weekly Assignments
20%	Quizzes
20%	Mid Term Exam
25%	Final Exam

Advanced COBOL Week 6

Assigned Reading

Overview

Important COBOL Stuff

Assignment

Assigned Reading

Horn and Gleason - Chapter 16 - Subprograms - Review

Additional 'Hot Topic' Reading on the Internet:

First, goto:

http://www.year2000.com/

Read their current featured article:

Year 2000: The MacGyver Solution by Lorin May

From the year 2000 Archive on that page read:

Party When It's 1999

Implementing a Site Review Program to Prepare Your Organization for Y2K and Other Risks by Kathleen Tudor, KPMG

Code for Year 2000 Date Routines -- by Raymond H. Thompson

Then, go to an article that discusses the costs of dealing with this problem:

The Real Costs of the Year 2000 Disaster by Paul A. Strassmann

http://www.strassmann.com/pubs/cw/real2000.shtml

Overview

We have some additional work to do to ensure that we clearly understand Subprogram linkage. You should, therefore, carefully review the material in Chapter 16 of your text. More about this in the **Important COBOL Stuff** that follows.

I'm also having you diverge a bit from your text work to do a little research on a very controversial COBOL topic - the Year2000 problem. You've likely heard something about it on the news or perhaps in some of your other classes. The articles selected above will give you a feel for the intensity of the problem, the logistics in planning the solution, the complexity of the coding solutions and the insurmountable cost to fix the problem. Realize as you read these articles that there are two ways to handle the problem -

- fix the data and expand the date fields in the programs that use the data
- modify only the programs to include temporarily code to deal with the problem.

When you read *Code for Year 2000 Date Routines -- by Raymond H. Thompson,* do not try to understand the code. Realize that this is temporary fix code for the Year2000 problem and that I just want you to see the complexity of dealing with date fix logic.

Important COBOL Stuff

Remember last week I told you that the purpose of using Subprograms is to isolate specific functions making the task of maintaining a large system easier and saving resources in that the programs can be re-used in other areas.

I also gave you the following terms and would like you to review them now before we move on:

Calling Program(Module) - a program that **passes control** to another program to perform a specific function.

Called Program(Module) - a program that **receives** control from another program to perform a specific function.

LINKAGE SECTION - a common area in the computers memory, reserved by the calling program, that is used to pass data values between programs.

CALL - the COBOL statement a Calling Program uses to execute a Subprogram

GOBACK - the COBOL statement a Called Program uses to return control to the Calling Program.

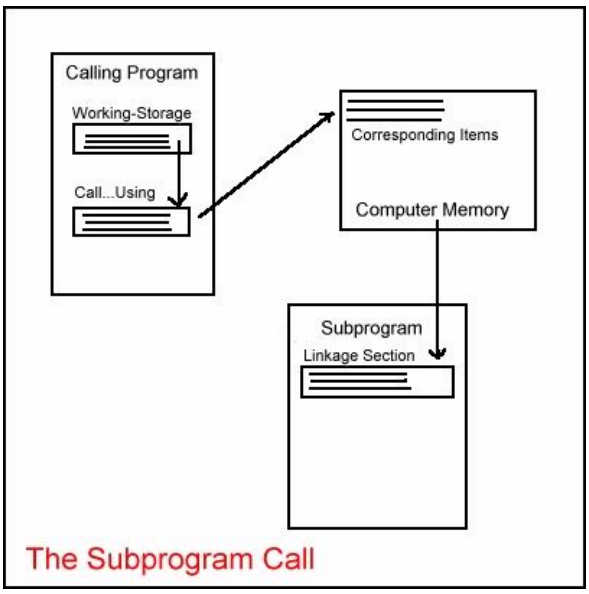
CALL ....Subprogram name.... USING - the COBOL statement used to both call (CALL) a Subprogram and set up the are in the computer's memory (USING) that will initialize the LINKAGE SECTION in the Called Program.

PROCEDURE DIVISION USING ...variable name(s).... the COBOL statement used in the Called Program to list the variables that the Called Program will work with and manipulate in the LINKAGE SECTION of the Subprogram.

With me so far??

OK, let's look a little closer at the logistics of how this works with some diagrams and follow the sequence via a step-by-step overview. Again, assume a simple scenario where an item of information, such as a social security number, is being looked-up based on a name search. Assume that the Calling Program has 2 Working-Storage fields, last-name, social-security-number.

Illustration A



1. The Calling Program has been invoked, is running in memory (Illustration A)

2. The operator, needing a client's social security number, is prompted for and enters a client Last Name (Illustration A)

3. The instruction to CALL the Subprogram is encountered (Illustration A)

```
IDENTIFICATION DIVISION.
PROGRAM NAME. PROGRAM1.
WORKING-STORAGE SECTION.
01 CUSTOMER-INFORMATION.
    05 LAST-NAME.
    05 SOCIAL-SECURITY-NUMBER.
01 OUTPUT-RECORD.
PROCEDURE DIVISION.
MOVE LAST-NAME TO OUTPUT-RECORD.
CALL 'PROGRAM2' USING LAST-NAME
                      SOCIAL-SECURITY NUMBER.
MOVE SOCIAL-SECURITY-NUMBER TO OUTPUT RECORD.
```

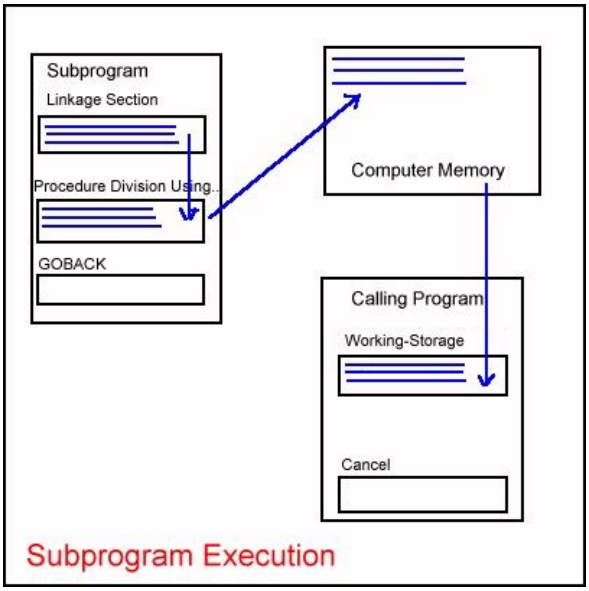
4. The Working-Storage fields that are stored in the calling program are mapped in the computer's memory via the Call....Using statement (Illustration

A)

5. Via the Linkage-Section, the Subprogram is given access to **all the fields named in the CALL statement** of the Calling Program (Illustration A)

6. The Subprogram, now also loaded into memory, begins to perform its function of searching a table based upon last name (Illustration B)

Illustration B



7. The Subprogram works only with those fields listed in the PROCEDURE DIVISION USING Statement. In our case, it would be both fields, last-name and social-security-number (Illustration B)

```
IDENTIFICATION DIVISION.
PROGRAM NAME. PROGRAM2.
WORKING-STORAGE SECTION.
01 LAST-NAME-TABLE.
    05 TABLE-LAST-NAME.
    05 TABLE-SS-NUMBER.
LINKAGE SECTION.
    05 LAST-NAME.
    05 SOCIAL-SECURITY-NUMBER.
PROCEDURE DIVISION USING LAST-NAME
                         SOCIAL-SECURITY NUMBER.
SEARCH LAST-NAME-TABLE FOR LAST-NAME
MOVE TABLE-SS-NUMBER TO SOCIAL-SECURITY-NUMBER
GOBACK.
```

8. When the search is ended, (a table hit), the social security number is moved, by the Subprogram to the field named social-security-number (Illustration B)

9. Having completed its functions, the GOBACK statement is encountered and control is passed back to the Calling Program (Illustration B)

10. The CANCEL Statement is encountered purging the Subprogram from memory (Illustration B)

11. The social-security-number, supplied by the Subprogram, is now in the Calling Program's Working-Storage and is displayed on-screen for the operator

---

Assignment

A. To demonstrate your understanding of Subprogram interaction, answer the following true/false questions and send them, via e-mail to your instructor. Realize that this is not a test and I'm just getting a feel for your comprehension. I want to see you attempt to answer these questions - that's what counts.....

1. All Working-Storage fields in the Calling Program must be used in the Call...Using Statement.

2. A Calling Program is only allowed to call one Subprogram during execution.

3. An executing Subprogram may Call another Subprogram.

4. In a large system, the call stream of Subprograms could be a hundred programs deep.

5. Two different Calling Programs can Call the same Subprogram at the same time.

6. A Subprogram can, in turn, Call the program that Called it.

7. Now think..... The order of fields in the **Call...Using**, the Subprogram **Linkage Section** and the Subprogram **Procedure Division Using...** must be in the same order.

8. A Subprogram can have its own Working-Storage fields apart from the Linkage Section fields.

9. The GOBACK statement is used to remove a program from memory.

10. The same Subprogram cannot be called twice from the same Calling Program.

12 Fill in the blank - When the Subprogram has control of the computer, the Calling Program is 'hanging in limbo'. Issuing a _____ immediately gives control back to the Calling Program.

B Download the following file: Programs for Week Six

Extract the files into a directory where you can access them with the Micro Focus Personal COBOL Editor. Don't even try to compile the programs - they are far from ready for prime time and I have purposely excluded a required Copybook and the data file. What I want you to do is 'desk check', (a term for visually inspecting code) the programs and **tell me**:

1. **the name** of the Calling program

2. **the name** of the Called Program (subprogram)

3. **if**, based on the way the Call...Using, Linkage Section and Procedure Division Using.... are coded, the programs will work....and....what needs to be done, if anything???????????

Advanced COBOL Week 10

Assigned Reading

Horn and Gleason - Chapter 13 - Sequential File Maintenance

Assigned Reading

Overview

Assignment

Overview

This week, we are covering a topic that is vital to your knowledge of COBOL programming. Sequential File Maintainance, also known as Master-File Update Processing, has a broad bandwidth of use throughout the IT industry.

While your text book is careful to treat changes to sequential files as separate topics, you should be aware that programs designed to handle Master File Update logic process all changes during program execution and actually function to produce a **New** Master File.

While used extensively in Tape Processing, the logic that you will learn is applicable to any situation where transcactions are accumulated and processed in groups, typically at the end of some business cycle (end of the day, perhaps). You will find these applications referred to as Batch Processing.

Typically:

- A New Master File is Created by Combining the Data in an Old Master File with a Transaction File
- A Transaction File Carries an Action Indicator:
  - A=Add
  - D=Delete
  - C=Change
- Requires the following files:
  - Input: Old Master File, Transaction File
  - Output: New Master File, Error Report
- Both the Old Master and Transaction File must:
  - Have a similar key
  - Be arranged in ascending order on the key
  - Have HIGH-VALUES (9999's) in the key field in the last record. The last record is known as a trailer record.

Below you will see a sample Old Master File, Sample Input File, Sample Program and Sample Error Report. See if you can walk through the code and follow the processof how the New Master File is built. As for the Error Report, you will want to follow its creation as well for the assignment that follows below.

Input File (Old Master File)	Input File (Transaction File)
001 PIPPLEWICK POPPY	A 001 PIMPPLEWICK POPPY
005 GRUMPER GALVIN	A 002 ICKELPICKLE ELSIE
006 ROPER ROBERT	A 003 WIMPPY WALLACE
007 STROMBERG STEVEN	C 004 WHIPLASH BUMB
008 FARLEY PHYLLIS	D 012 SEVILLE BARBARA
009 STERLING RONALD	X 013 PEACE WARREN
010 DATE JULIAN	A 016 QUITO AMOS
011 WESTMORELAND LEE	A 022 GUSTAFFSON HARRY
012 SEVILLE BARBARA	C 023 LESSE MOIRA
013 PEACE WARREN	D 024 YANG CYNTHIA
014 MELLOW MARCIA	C 025 WHELAN ROD
015 STEWART JANET	A 026 JOHNSON SARAH
999 TRAILER RECORD	999 TRAILER RECORD

Output File (New Master File)	Output File (Error File)
001 PIPPLEWICK POPPY 002 ICKELPICKLE ELSIE 003 WIMPPY WALLACE 005 GRUMPER GALVIN 006 ROPER ROBERT 007 STROMBERG STEVEN 008 FARLEY PHYLLIS 009 STERLING RONALD 010 DATE JULIAN 011 WESTMORELAND LEE 013 PEACE WARREN 014 MELLOW MARCIA 015 STEWART JANET 016 QUITO AMOS 022 GUSTAFFSON HARRY 026 JOHNSON SARAH	A 001 PIMPPLEWICK POPPY C 004 WHIPLASH BUMB X 013 PEACE WARREN C 023 LESSE MOIRA D 024 YANG CYNTHIA C 025 WHELAN ROD

Sample Master-File Update Routine

```
200-MAINLINE-ROUTINE.
IF M-CODE = T-CODE
PERFORM 300-PROCESS-CHANGE-ROUTINE
ELSE
IF T-CODE < M-CODE
PERFORM 350-PROCESS-ADD-ROUTINE
ELSE
PERFORM 375-PROCESS-MAST-ROUTINE
PERFORM 850-READ-OLD-MAST-ROUTINE.

300-PROCESS-CHANGE-ROUTINE.
IF T-ACTION-CODE = 'C'
PERFORM 395-PROCESS-TRANS-ROUTINE
PERFORM 850-READ-OLD-MAST-ROUTINE
PERFORM 860-READ-TRX-ROUTINE
ELSE
IF T-ACTION-CODE = 'A'
PERFORM 500-PROCESS-ERROR-ROUTINE
PERFORM 860-READ-TRX-ROUTINE
ELSE
IF T-ACTION-CODE = 'D'
PERFORM 850-READ-OLD-MAST-ROUTINE
PERFORM 860-READ-TRX-ROUTINE
ELSE
PERFORM 500-PROCESS-ERROR-ROUTINE
PERFORM 860-READ-TRX-ROUTINE.

350-PROCESS-ADD-ROUTINE.
IF T-ACTION-CODE = 'A'
PERFORM 395-PROCESS-TRANS-ROUTINE
ELSE
PERFORM 500-PROCESS-ERROR-ROUTINE.
PERFORM 860-READ-TRX-ROUTINE.
```

Assignment

Be sure you read through this material really well. Next week you will have a programming assignment involving sequential file updates.

For this week, I want you to look above at the sample error report. Send me a list of each of the errors listed and tell me **why** they are error records.

Advanced COBOL Week 14

Assigned Reading - Review

Horn and Gleason - Chapter 14 - Indexed Files

Horn and Gleason - Chapter 15 - Relative Files

Assigned Reading

Overview

Assignment

Overview

File Organization

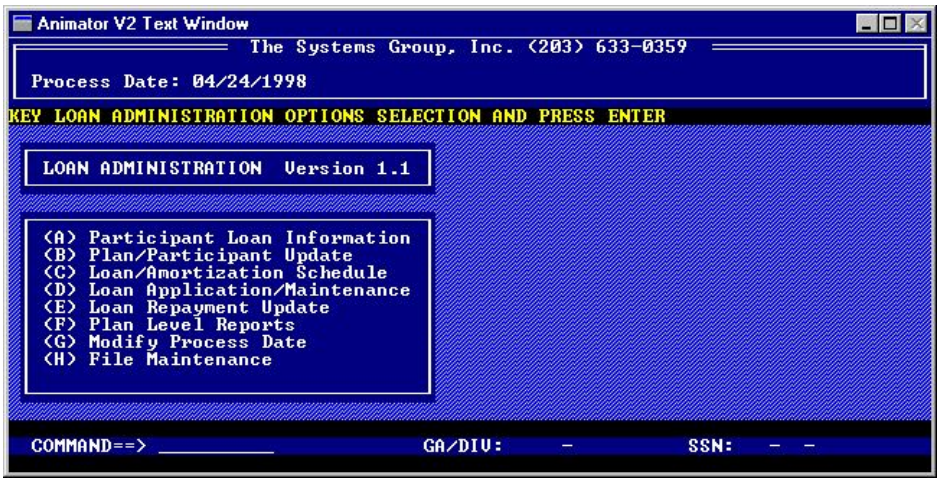
Earlier in the semester (week 2) you downloaded a group of programs that comprised a portion of a PC Loan Administration System. Nearly all the files used in this system are indexed files.

A sample SELECT from one of the programs appears as follows:

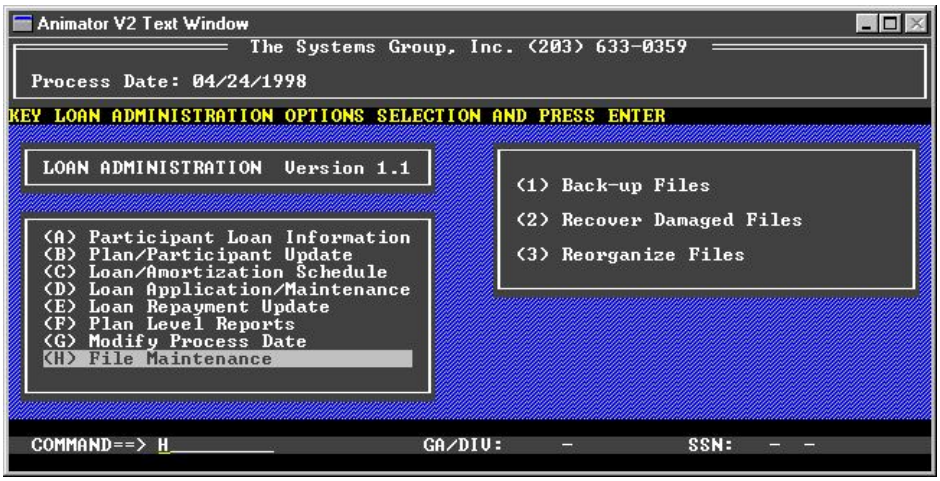
```
SELECT OPTIONAL LNFILE ASSIGN TO "LOANFILE.DAT"
ORGANIZATION IS INDEXED
ACCESS MODE IS DYNAMIC
RECORD KEY IS LN-LOAN-KEY
FILE STATUS IS LN-FILE-STAT.
```

Currently, all SELECT statements are coded properly and if you were asked to add files into the system, you could simply clone the above statements and fold them into a new SELECT statement with minor changes.

Recall, that a user never sees files names, doesn't know if the file ORGANIZATION is RANDOM or INDEXED, whether the programmer has specified DYNAMIC ACCESS, etc. All their file access needs are handled by the program. They key in a COMMAND, a DIVISION NUMBER and a SOCIAL SECURITY NUMBER and the loan information for that person is displayed. Remember the screen below?



Take a close look at the menu options provided to the USER of the system. Look at the last option **[H] File Maintenance**. What do you suppose this option is for? If you were to run the existing code you have right now, this is what you would see:



Note that selecting option **[H] File Maintenance** opens a new menu giving the options to back-up, recover and reorganize files. When you realize that this system was designed to run stand-alone at individual branch offices, the need for these utilities becomes clear. Each branch, while using files with the same names will have different records in the files that match their accounts. Therefore, a method is provided that automates:

1. file back-up
2. file recovery to rebuild file indexes in the event they become damaged due to power outages, surges, etc.
3. file reorganization to restore files from back-ups.

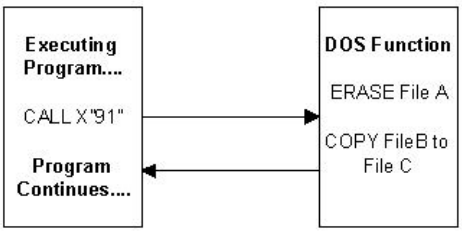
With me so far?? Good.

Assignment

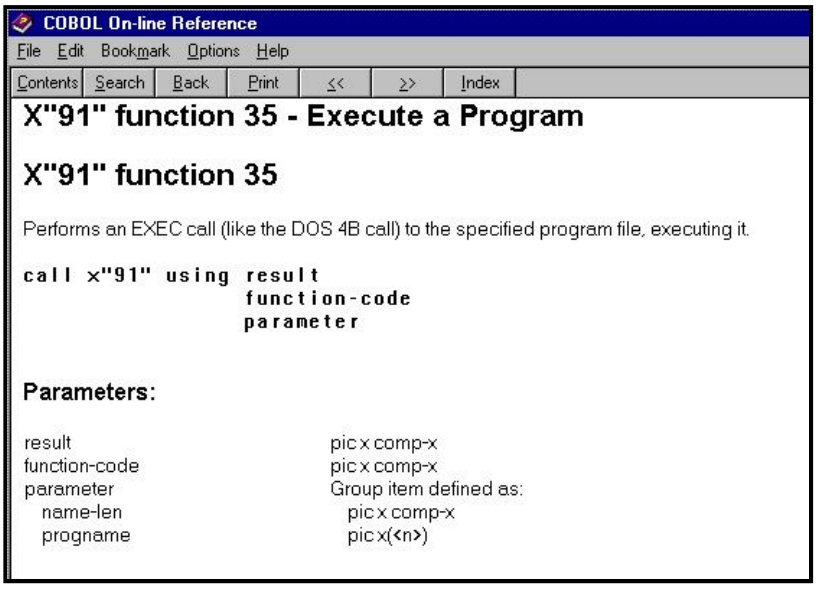
This assignment is meant to give you a real world experience with file maintenance. What you have here is a system that worked fine under DOS and Windows 3.1. Along came Windows 95 and the nightmare was about to become reality until you, an expert programmer came along to save the day. The current system is written in Micro Focus COBOL and the backup-up, restore and recover utilities are all set up to invoke Micro Focus DOS command line utilities. So, first a word about how that works.

Micro Focus COBOL allows a programmer to invoke DOS commands from a program while the program is executing. In Micro Focus and Microsoft terms, this is referred to as 'invoking command from the DOS shell.

It is accomplished via a DOS-CALL, a CALL to execute a function that's normally done in DOS like copying a file, erasing a file, using the DIR command, etc. It is accomplished by using one of the many Micro Focus HEX call functions that the developer has available, in this case a HEX 91 call that is coded as 'CALL X"91". This give momentary control over to DOS to execute whatever DOS command the programmer wishes to invoke as diagrammed below:



There are additional parameters required when using the Hex 91 CALL. Executing a DOS command is one of the functions of Hex 91, specifically Function 35. Here's what a portion of the Micro Focus On-line help looks like for this CALL:


COBOL On-line Reference

File Edit Bookmark Options Help

Contents Search Back Print << >> Index

X"91" function 35 - Execute a Program

X"91" function 35

Performs an EXEC call (like the DOS 4B call) to the specified program file, executing it.

```
call x"91" using result
                 function-code
                 parameter
```

Parameters:

result	pic x comp-x
function-code	pic x comp-x
parameter	Group item defined as:
name-len	pic x comp-x
progname	pic x(<n>)

Here's what the code looks like being used in the LAS System:

```
WORKING-STORAGE SECTION.
01 RES-ULT PIC 99 COMP.
01 FUNCTION-35 PIC 99 COMP VALUE 35.
01 CALL-PARM.
  05 FILLER PIC 99 COMP VALUE 0.
  05 FILLER PIC X.

PROCEDURE DIVISION
display "ECHO INSTALL PLANFILE BACK-UP DISK IN DRIVE A" upon command-
line
PERFORM 710-DOS-CALL
display "PAUSE" upon command-line
PERFORM 710-DOS-CALL
display "COPY PLANFILE.* A:" upon command-line
PERFORM 710-DOS-CALL.

710-DOS-CALL.
CALL X"91" USING RES-ULT
                 FUNCTION-35
                 CALL-PARM.
```

Let's follow what's in progress in this program....

1. The required parameters are coded in **Working-Storage** as required. Result, Function and Call Parm are required fields used by the Hex CALL and the only one we need to assign a value to is the Function field where we specify Function **35** via a value clause. The other fields are used internally by the CALL, so, we don't assign a value to them.
2. At the start of the **Procedure Division**, a **display** command is used to specify the **DOS command** we want to execute. As you look at the code, visualize what is going on here:
   - The DOS ECHO command is set up to be issued. This displays the associated message which will ask the user to insert a floppy disk in drive A.
   - The ECHO command is issued when 710-DOS-CALL is executed.
   - Control returns to the program.
   - The DOS PAUSE command is set up to be issued with a display statement.
   - The PAUSE command is issued when 710-DOS-CALL is executed.
   - When the PAUSE is issued, the message 'strike any key to continue .....' is displayed by the pause command giving the user time to insert their A disk into the A drive.
   - When the user strikes a key, control returns to the program.
   - The DOS COPY command is set up to be issued with a display statement.
   - In this case all files with the name 'PLANFILE', (PLANFILE.*) are copied to the A drive when 710-DOS-CALL is executed.

- Control is returned to the program.

Here's what the user's screen would look like as this program executes.....and remember, this DOS screen or DOS shell displays when, from the program interface, the user has selected option [1] Back-up Files from the submenu displayed after choosing [H] File Maintenance.....

```
C:>INSERT PLANFILE BACK-UP DISK IN DRIVE A
C:>strike any key to continue. . .
```

Then, when the user has inserted the diskette and struck a key, the screen would appear as follows.....

```
C:>INSERT PLANFILE BACK-UP DISK IN DRIVE A
C:>strike any key to continue. . .
C:>COPY PLANFILE.* A:
```

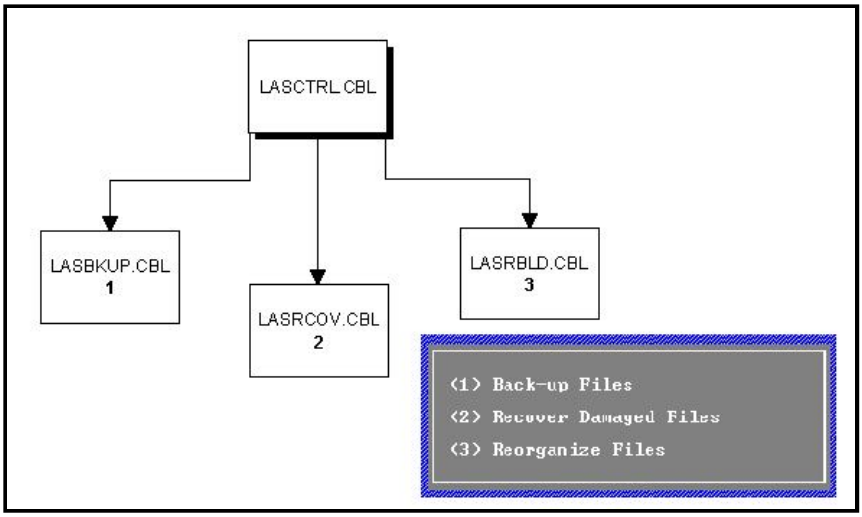
If the actual program were allowed to continue, the following would be the result:

```
C:>INSERT PLANFILE BACK-UP DISK IN DRIVE A
C:>strike any key to continue. . .
C:>COPY PLANFILE.* A:


C:>INSERT LOANFILE BACK-UP DISK IN DRIVE A
C:>strike any key to continue. . .
C:>COPY LOANFILE.* A:


C:>INSERT PLPTFILE BACK-UP DISK IN DRIVE A
C:>strike any key to continue. . .
C:>COPY PLPTFILE.* A:
```

Yup, all that code to back up their files and picture the user flipping floppies in and out of the drive!!

Well, that code was written to do file maintenance in 1991 and was surely a state-of-the-art methodology. However with today's liberal hard drives and network storage, this system will have to be upgraded to simplify file maintenance.

First, here's the flow chart for how the system is currently set-up:



Each file maintenance submenu option invokes a program. Your job will be to upgrade these programs so that:

1. The files can be backed-up to the C: drive.
2. Damaged Indexes can be recovered.
3. The files can be restored from the C: drive.

To add to your problems, the above coding scheme is no longer supported under Windows 95!! In order to execute DOS commands from the DOS shell you must now build executable files, also referred to as batch files.

So.....the above Procedure Division code that looked like this.....

```
PROCEDURE DIVISION
display "ECHO INSERT PLANFILE BACK-UP DISK IN DRIVE A" upon command-line
PERFORM 710-DOS-CALL
display "PAUSE" upon command-line
PERFORM 710-DOS-CALL
display "COPY PLANFILE.* A:" upon command-line
PERFORM 710-DOS-CALL.
```

can/must be replaced by this for each backup/recover/restore in the programs.....

```
PROCEDURE DIVISION
display "PARTFILE.BAT" upon command-line
PERFORM 710-DOS-CALL.
```

You create the batch file using an editor like Notepad or you can use the Micro Focus Editor saving the file in your program subdirectory with a unique filename such as PARTFILE.BAT, LOANFILE.BAT, etc., and here's what the batch file looks like.....

```
echo off
echo Backing up PARTFILE.dat and PARTFILE.IDX to C:\LAS
copy partfile.DAT c:\las
copy partfile.idx c:\las
pause
```

Additionally, you can string all of your files into one batch file....

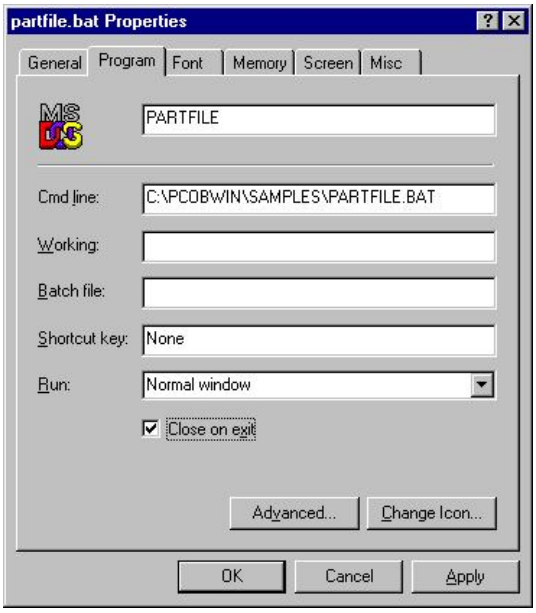
```
echo off
echo Backing up Loan Administration Files to C:\LAS
copy partfile.DAT c:\las
copy partfile.idx c:\las
copy loanfile.DAT c:\las
copy loanfile.idx c:\las
pause
```

A couple of final notes before you get into the specifics of the assignment.

The Micro Focus REBUILD.EXE file is used to fix corrupted index files. It inspects the data file to be rebuilt and rewrites a new index. This utility file is not supplied with Micro Focus Personal COBOL, so, you will have to download it below.

Control will not automatically return to your program after a batch file is executed in the Windows 95 DOS shell as it did in Windows 3.1. The user actually would have to close the window created and this is not really desirable. You want that Window to close automatically. So, here's what you have to do. Once you have created a batch file (a .BAT):

1. Start up Windows Explorer.
2. Find your batch file and right click on it.
3. Select Properties
4. Click on the Program Tab and check 'Close on exit'
5. Click OK.



This will actually create an additional file to accompany the .BAT file. It will be a .PIF file with the same file name.

Assignment Specifics - Check List - Suggest Approach.....

☐	Check to see if you still have the programs that you downloaded Week 2 and worked on in Week 3. If not, download them again and put them all in your default work PCOBWIN work directory (usually \PCOBWIN\SAMPLES).
☐	Of those programs, LASCTRL.CBL, P249COPY.CBL and LASMENU.CBL must be compiled clean to test your new/fixed programs.
☐	Download FILMAINT.EXE. When expanded in your work directory you should have 3 files: • LASRCOV.CBL • LASRBLD.CBL • LASBKUP.CBL.
☐	Download REBUILD.EXE and put it in your work directory. Do not try to expand or execute this file. This is a Micro Focus utility that one of your programs will call.
☐	Compile the three 1991 antique programs to ensure that they compile clean and to ensure that you have all required copybooks.
☐	Check your execution logic by opening LASCRL.INT for execution. Run the program, select [H] File Maintenance, [1] Back-up Files and press the Enter key when prompted. It should open a do-nothing DOS window which you can close and then suspend animation of the program.
☐	Using the format above for stringing all files into one batch file, create a file to back up the files that are being backed up in BACKUP.CBL. Name your new file BKUP.BAT . This .BAT file should be in your work directory.
☐	Using the format above for stringing all files into one batch file, create a file to reindex the files that are being reindexed in LASRCOV.CBL. Be sure to use the same format of the REBUILD command (REBUILD FILENAME.DAT /E /I /V). Name your new file RCOV.BAT. This .BAT file should be in your work directory.
☐	Using the format above for stringing all files into one batch file, create a file to restore from backup the files that are being restored in LASRBLD.CBL. Name your new file RBLD.BAT . This .BAT file should be in your work directory.
☐	Create a .PIF file for each of the above .BAT files so they will automatically close when finished. These .PIF files should be in your work directory.
☐	Look at each of the three programs carefully. Remove all the antique CALL's to 710-DOS-CALL and the associated display commands. Replace the removed code with the sample code shown above for executing a .BAT file. Be sure to

	specify the proper .BAT file for each program.
☐	Recompile your three file maintenance programs.
☐	Create a new subdirectory called C:\LAS.
☐	Animate LASCTRL and test [1] File Back-up.
☐	Inspect the C:\LAS subdirectory. You should have copies of all the files and their indexes in that subdirectory.
☐	Animate LASCTRL and test [2] Recover Damaged Files.
☐	Inspect your work directory (\PCOBWIN\SAMPLES). Your index files should have a new time-stamp on them with today's date.
☐	Animate LASCTRL and test [3] Reorganize Files.
☐	Inspect your work directory (\PCOBWIN\SAMPLES). Your index files should have the old time-stamp on them from the backup directory.

When you have finished the assignment, send the following files to your instructor:

- LASRCOV.CBL
- LASRBLD.CBL
- LASBKUP.CBL
- RCOV.BAT
- RBLD.BAT
- BKUP.BAT
- RCOV.PIF
- RBLD.PIF
- BKUP.PIF.
