Shell Programming - Points: 70 (11 of 35%)

[TEST PART II - CLOSED BOOK - NO COMPUTER] Write code for an executable shell script that will do the following actions, in the exact order given below. You will write approximately 40-45 lines of executable code. For full marks, you must put a one-line comment containing the step number in front of the executable code in each step. Do not put a Step Number comment as the first line of the script!

Summary and Purpose (what this script will do):

> The script expects a single optional C++ program source file name on the command line. If the file argument is missing, get the file name from the user. It will compare the number of lines in this source file with the number of lines in another file, and it will compile whichever program has more lines. It backs up the output file first.

---

1. *[Points: 5]* Structure your script to include these standard DAT2330 script sections: interpreter, one-line description, syntax, search path, and file creation mask. Do *not* include the Purpose or Assignment Label sections.
2. *[Points: 7]* Issue a good error message (follow the DAT2330 guidelines) and exit the script with status 1 if there is more than one command line argument.
3. *[Points: 7]* If there is a command line argument, put it into a variable named **`prog`** ; otherwise, the script must get the C++ program source file name from the user and put it into the same named variable. Quick-exit the script with status 2 if the user signals EOF to the script. (No error message is needed.)
4. *[Points: 6]* Make sure the pathname in variable **`prog`** is not an empty string; otherwise, issue a good error message and exit the script with status 3.
5. *[Points: 7]* Make sure the pathname in variable **`prog`** is a plain file; otherwise, issue a good error message and exit the script with status 4.
6. *[Points: 7]* Make sure the file whose name is in variable **`prog`** is not an empty file (has a size larger than zero); otherwise, issue a good error message and exit the script with status 5.
7. *[Points: 6]* If the file whose name is in variable **`prog`** is not readable, add read permissions for user and group (not for others). Quick-exit the script with status 6 if changing permissions fails. (No output or error messages are needed.)
8. *[Points: 4]* Put only the count of lines in the file named by the **`prog`** variable into a variable named **`plines`** . Quick-exit the script with status 7 if the counting fails.
9. *[Points: 4]* Put only the count of lines in the file named **`foo.cpp`** into a variable named **`flines`** . (You may assume this file exists; no code is needed.) Quick-exit the script with status 8 if the counting fails.
10. *[Points: 4]* If something with the name of **`happy`** already exists in the current directory, rename it to be **`happy.bak`** .
11. *[Points: 8]* Compile whichever of the two C++ source files that has the most lines into an output file named **`happy`** . Quick-exit the script with status 9 if the compile fails.
12. *[Points: 5]* If there is a readable **`Makefile`** in the current directory, and if the **`Makefile`** contains the name of the larger of the two C++ source files, print a message on standard output telling the user to use the **`make`** program to compile the file next time. The message must contain the name of the source file. No other output should appear in this step.

Put a one-line comment containing the step number on the line above the executable code in each step.