Shell Script - Points: 70 (10% of 30%)

Write code for an executable shell script that will do the following actions, in the exact order given below. (You will write approximately 40 lines of executable code, plus a **step number** comment before each step.)

Summary and Purpose:

The first argument to the script is a C++ program source file name. The second argument is an optional output file name. (The script will ask for a missing second argument.) The script will compile the source file into the output file and output a message about the relative sizes of the input and output files.

Comments Required:

Put a one-line comment containing the **step number** in front of each step.

---

1. *[Points: 5]* Start your script with all the parts of a correct DAT2330 script header; however, do *not* include the Purpose or Assignment Label. (Remember the One-Line Description and Syntax.)
2. *[Points: 8]* If the number of arguments is less than 1 or greater than 2, issue a good error message (follow the DAT2330 guidelines) and exit the script with status 1.
3. *[Points: 2]* Put the first argument (the C++ source file name) into a variable named **`src`**.
4. *[Points: 7]* Make sure the pathname in variable **`src`** is a plain file; otherwise, issue a good error message and exit the script with status 2.
5. *[Points: 7]* Make sure the file whose name is in variable **`src`** is not an empty file (has a size larger than zero); otherwise, issue a good error message about the file size and exit the script with status 3.
6. *[Points: 8]* If there are two arguments, put the second argument (the output file name) into a variable named **`out`**; otherwise, get the missing output file name from the user. Put the output file name entered by the user into variable **`out`**. Quick-exit the script with status 4 if the user signals EOF to the script. (No error message is needed.)
7. *[Points: 6]* Make sure the name contained in the variable **`out`** is not an empty (null) string; otherwise, issue a good error message about the null input string and exit the script with status 5.
8. *[Points: 4]* Add read permissions for group (not for user or others) to the the file whose name is in variable **`src`**. Quick-exit the script (no message) with status 6 if changing permissions fails.
9. *[Points: 4]* If the file whose name is in variable **`out`** already exists, rename it to be **`backup.bak`**.
10. *[Points: 4]* Compile the source file into the output file. Enable all warnings during the compile.
11. *[Points: 4]* Put the count of characters contained in the source file into a variable named **`schars`**.
12. *[Points: 4]* Put the count of characters contained in the output file into a variable named **`ochars`**.
13. *[Points: 7]* Compare the sizes of the source and output files and display **one** of the following messages:

```
source XXX is larger than output YYY
source XXX is the same size as output YYY
source XXX is smaller than output YYY
```

where **`XXX`** is replaced in your output by the actual name of the source file and **`YYY`** is replaced by the actual name of the output file. Do not output the strings **`XXX`** or **`YYY`**. Output only **one** of the messages.

Put a one-line comment containing the step number in front of each step.

− FIN −