1 Setup:

In this final exam, you will create 2 BASH scripts and 1 Makefile to create a coding environment. All C++ code has been provided on Dropbox under the "Final Exam" assignment as 'code.tar.gz'.

1. Start in your home directory.
2. Create a directory inside your Desktop directory called 'username_final' with your username and an underscore between your username and the word 'final'.
3. ***(1 pt.)*** Download the 'code.tar.gz' file from Dropbox under "Final Exam" and move it into the 'username_final' directory you created on your Desktop.
4. ***(1 pt.)*** Use the 'tar' command to extract the 'code.tar.gz' files and directories. If done correctly, you should see:

```
code/
code/src/
code/src/main.cpp
code/src/fib.cpp
code/inc/
code/inc/main.h
code/inc/fib.h
```

5. ***(1 pt.)*** Move the 'src' and 'inc' directory from inside the 'code' directory to the current directory (i.e. inside your 'username_final' directory).
6. ***(1 pt.)*** Delete the 'code.tar.gz' file and the leftover 'code' directory.
7. ***(1 pt.)*** Create two files called 'a_submit.sh' and 'b_extract.sh'.
8. ***(1 pt.)*** Change the permissions of all the '.sh' files you just created to ONLY grant yourself read, write, and execute permissions. If you grant anyone else permissions, you may lose points.
9. Create a file called 'Makefile'.

10. Your file/directory structure should look like the following when you run the 'tree' command:

```
.
|-- a_submit.sh
|-- b_extract.sh
|-- inc
|   |-- fib.h
|   |-- main.h
|-- Makefile
|-- src
    |-- fib.cpp
    |-- main.cpp

2 directories, 7 files
```

2 Makefile

Open your 'Makefile' script for editing using a command line text editor. Your Makefile should contain the following:

1. ***(.5 pt.)*** Create the following variables **at the top** of your Makefile.

```
CC=g++
CFLAGS=--std=c++17
EXE=fib
SRC=./src
INC=./inc
OBJ=./obj
DEBUG=./debug.log
```

2. ***(.5 pt.)*** Create the 'all' target below the variables you just assigned. It should depend on the 'clean', 'obj', 'fib.o', and 'main.o' targets/files/directories and should run the following command:

```
$(CC) $(CFLAGS) -o $(EXE) $(OBJ)/*
```

3. ***(.5 pt.)*** Create the 'main.o' target below the 'all' target you just defined. It should depend on the 'obj', '$(INC)/main.h', and '$(SRC)/main.cpp' targets/files/directories and should run the following command:

```
$(CC) $(CFLAGS) -c $(SRC)/main.cpp -o $(OBJ)/main.o
```

4. ***(.5 pt.)*** Create the 'fib.o' target below the 'main.o' target you just defined. It should depend on the 'obj', '$(INC)/fib.h', and '$(SRC)/fib.cpp' targets/files/directories and should run the following command:

```
$(CC) $(CFLAGS) -c $(SRC)/fib.cpp -o $(OBJ)/fib.o
```

5. ***(1 pt.)*** Create the 'obj' target below the 'fib.o' target definition and have it create a directory called 'obj' using the option that does not throw an error if the directory already exists.

6. ***(1 pt.)*** Create the 'submit' target below the 'obj' target definition and have it run the 'a_submit.sh' script.

7. ***(1 pt.)*** Create the 'extract' target below the 'submit' target definition and have it run the 'b_extract.sh' script.

8. ***(1 pt.)*** Create the 'test' target below the 'extract' target definition and have it run the executable by simply running './$(EXE)'.

9. ***(.5 pt.)*** Create the 'clean' target below the 'test' target definition and have it run a command to recursively and forcibly delete the files/directories associated with the 'EXE', 'OBJ', and 'DEBUG' variables. (Hint: You need to access the variable. Accessing variables in a Makefile is not the same as in shell scripts!)

10. ***(.5 pt.)*** Create the 'cleanall' target below the 'clean' target definition and have it depend on the 'clean' target and run a command that deletes the 'testing', 'submissions', and 'out' directories. We will create these directories later. No need to create them in the Makefile. (Hint: You must recursively and forcibly delete these directories.)

You should use your Makefile to test your scripts.

3 ./a_submit.sh

Do NOT use 'cd' or &&! Hint: Use a $ in front of variable names to access the value stored in the variable. Your a_submit.sh script should contain:

1. The shebang as the top line. Be sure not to miss-type this part!
2. ***(.5 pt.)*** Assign a variable 'SUB' to './submissions'.
3. ***(.5 pt.)*** Assign a variable 'CURR' to the output of the 'date +%Y%m%d_%H%M%S' command. This command creates a string with the current date and time separated by an underscore.
4. ***(1 pt.)*** Create a directory inside the current directory using the value stored in the SUB variable. Be sure to use the option/flag that ensures creating the directory does not throw an error.
5. ***(1 pt.)*** Do NOT use 'cd'! Create a directory inside of the submissions directory named the value of CURR variable. Be sure to use the option/flag that ensures creating the directory does not throw an error.
6. ***(1 pt.)*** Recursively copy your Makefile, inc, and src file/directories to the CURR named directory inside of the SUB directory.
7. ***(1 pt.)*** Use the tar command to compress your CURR directory into a file named $CURR.tar.gz. Make sure that the resulting .targ.gz file is inside of the submissions directory (You can move it OR use the special tar option -C to help).
8. ***(1 pt.)*** Print the value of the CURR variable and **overwrite** a file called 'newest.log' in the 'submissions' directory.
9. ***(1 pt.)*** Recursively delete the CURR directory that is left over in the SUB directory.

Running your 'a_submit.sh' script using './a_submit.sh' should result in a 'submissions' directory that contains the 'newest.log' file and with a tar.gz file with the date and time as the name (i.e. 20230418_112805.tar.gz). Use 'tree submissions'.

```
submissions
|-- 20230418_112805.tar.gz
|-- newest.log
```

4 ./b_extract.sh

Do NOT use 'cd' or &&! Hint: Use a $ in front of variable names to access the value stored in the variable. Your b_extract.sh script should contain:

1. The shebang as the top line. Be sure not to miss-type this part!
2. Assign a variable named 'SUB' to './submissions'.
3. ***(1 pt.)*** Assign a variable named 'NEWLOG' to '$SUB/newest.log'.
4. ***(1 pt.)*** Use an if-statement to check if the submissions directory **does not** exist. If it doesn't, throw a message and use the 'exit' command to exit the script.
5. ***(1 pt.)*** Use an if-statement to check if the NEWLOG file **does not** exist. If it doesn't, throw a message and use the 'exit' command to exit the script.
6. ***(1 pt.)*** Assign a variable named 'NEWEST' to the output of a command that grabs the first line of the NEWLOG file.
7. ***(1 pt.)*** Assign a variable named 'NEWTAR' to '$SUB/$NEWEST.tar.gz'
8. ***(1 pt.)*** Use an if-statement to check if the NEWTAR file **does not** exist. If it doesn't, throw a message and use the 'exit' command to exit the script.
9. ***(1 pt.)*** Create a directory named 'testing' in the current directory using an option/flag that will not throw an error if it already exists. Use the tar command to extract the NEWTAR file into the 'testing' directory you just created.

Running your 'b_extract.sh' script after running your 'a_submit.sh' script should result in a 'testing' directory that contains a data/time named directory with a copy of the Makefile, inc, and src directories. Use 'tree testing'.

```
testing
|-- 20230418_112805
    |-- inc
    |   |-- fib.h
    |   |-- main.h
    |-- Makefile
    |-- src
        |-- fib.cpp
        |-- main.cpp
```

5 Submission:

You should remove the 'submissions' and 'testing' directories before submitting. You can use commands to do this or run 'make cleanall' assuming that your Makefile is correct.

Use the 'tar' command to tarball your username_final directory into username_final.tar.gz where username is your username.Your 'tar' command should output the following:

```
username_final/
username_final/a_submit.sh
username_final/b_extract.sh
username_final/src/
username_final/src/main.cpp
username_final/src/fib.cpp
username_final/inc/
username_final/inc/main.h
username_final/inc/fib.h
username_final/Makefile
```

You should NOT submit your username_final.tar.gz with any extra files or directories. The output of the 'tar' command should contain the exact same lines as below. However, it is OK if your lines are swapped around as long as the exact lines (no more and no less) are present.

Finally, submit your username_final.tar.gz file on Dropbox.

Download your submitted username_final.tar.gz file and check that it has all of the files/directories you expect it to!