CS270 Recitation 6
"Makefile Build Exercise"

Goals

1. To understand how to use a Makefile to build C source code into a program.
2. To learn how to modify the Makefile for new source files and targets.
3. To write yet another C program, this time using a data structure.

The Assignment

Make a subdirectory called R6 for the recitation, all files should reside in this subdirectory. Copy the following files to the R6 directory:

- http://www.cs.colostate.edu/~cs270/CurrentSemester/recitations/R6/struct.c
- http://www.cs.colostate.edu/~cs270/CurrentSemester/recitations/R6/struct.h
- http://www.cs.colostate.edu/~cs270/CurrentSemester/recitations/R6/main.c
- http://www.cs.colostate.edu/~cs270/CurrentSemester/recitations/R6/Makefile

1) Build the source code into an executable by typing the **make** comman, and note which commands are executed when the program is built. Edit the Makefile to see the executable name and run the program from the command line.

2) Extend the C *struct* called Student in the header file to add a last name, average homework score, average lab score, a midterm score, and a final exam score. All scores are of type integer in the range 0..100, and all names are character pointers (C strings).

3) Finish implementing the following functions in struct.c.

```
void inputScores(Student *student);
void outputScores(Student student);
```

The inputScores function should query standard input (with prompting) for the names and all scores. The outputScores function should print the names and scores (with labels) to standard output. Declare a global array of 5 students in main.c, and add a loop to the main entry point to input and output 5 student records.

4) Build the program again by typing the 'make' command, and iterate until the program builds and you can input and output student records. Hint: you can hit ctrl-C so that you don't have to enter more than a couple student records.

5) Add totalPoints (float) and finalGrade (char) fields to the structure in struct.h. Rebuild the program using the Makefile and notice which files are recompiled and linked.

6) Add a declaration and implementation of the following function to the header file:

```
void calculateScores(Student *student);
```

The calculateScores function will compute totalPoints and letterGrade according to the formula:

totalPoints = (homework average * 0.30) + (lab average * 0.20) + (midterm score * 0.20) + (final score * 0.30);

```
if (totalPoints > 90.0) letterGrade = 'A';
else if (totalPoints > 80.0) letterGrade = 'B';
else if (totalPoints > 70.0) letterGrade = 'C';
else if (totalPoints > 60.0) letterGrade = 'D';
else letterGrade = 'F';
```

You will also need to add print statements for totalPoints and letterGrade to the outputScores function.

7) Rebuild the program using the make command, and note which files are recompiled and linked. Test the program until it works, using the make command to rebuild. Touch main.c and rebuild the program using the make command, and note

which files are recompiled and linked. This is the purpose of a Makefile, to figure out based on dependencies which modules need to be rebuilt, without always building everything.

8) Type the following command to clean up the directory, and note which files are cleaned. Change the clean target to remove the R6.tar file.

$ make clean

9) Type the following command to package the directory, and note how the package is built.

$ make package

10) Show your working program and tar file to the TA, and explain what the three different targets in the Makefile are for: default, clean, and package. Submit to the drop box on Canvas for Recitation 6.