CS270 Recitation 4
"C Structures"

Goals

1. To write a C program from scratch, starting with only a header file.
2. To learn how to allocate and access C structs, and pass them around as parameters.
3. To write yet another C program, this time with dynamic allocation.

The Assignment

Make a subdirectory called R4 for the recitation, all files should reside in this subdirectory. Copy the following files to the R4 directory:

- http://www.cs.colostate.edu/~cs270/.Spring16/recitations/R4/struct.c
- http://www.cs.colostate.edu/~cs270/.Spring16/recitations/R4/struct.h
- http://www.cs.colostate.edu/~cs270/.Spring16/recitations/R4/main.c
- http://www.cs.colostate.edu/~cs270/.Spring16/recitations/R4/Makefile

1) 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 statically allocated character arrays (C strings) with 80 characters.

2) In main.c, declare a global pointer to a Student struct, outside of any functions. In the main entry point, print the prompt "Enter the number of students: " and read an integer from the user.

3) Again in the main entry point, dynamically allocate an array of Student structs based on the number entered. Add a statement to free the array at the end of the main entry point.

4) Implement 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.

5) Add a loop to the main entry point to input and output the specified number of student records.

6) 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.

7) 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.

8) 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 (with 2 digits after the decimal point) and letterGrade to the outputScores function.

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 the R4.tar package to the drop box on Canvas for R4.