CS 113
Homework Assignment 1

Given: September 03, 2009 **Due:** September 29, 2009

The assignment is due by 11:59 p.m. of the due date. The point value of each problem is shown in [ ]. Each solution must be the student's own work. Assistance should only be sought or accepted from the course instructor. Any violation of this rule will be dealt with harshly. Follow the documentation guidelines while writing and documenting your programs. Programs that are not well-documented will be penalized heavily. Your program must work on *all* possible inputs.

[**120**] **1. Grade Statistics** Suppose that you have been hired by a professor at a university to write a software that would help an instructor for a course to organize the scores of all students and generate some statistics. The specifications for your program are as follows.

Input The input to the program will come from a file called "grades.in". This file will contain all the information relevant to the course. The format of the input file is described below. Let `<bs>` denote some (zero or more) number of white spaces and `<`*n* `bs>` denote *n* white spaces. Let `<bl>` denote some number of blank lines and `<`*n* `bl>` denote *n* blank lines.

```
Class<bs>:<bs><course number>
Semester<bs>:<bs><year info>
Institute<bs>:<bs><university>
Instructor<bs>:<bs><instructor>
Title<bs>:<bs><title>
<bl>
Homeworks:<bs><number of hws><bs> , <bs><weightage of homeworks>
Quiz:<bs><number of quizzes><bs> , <bs><weightage of quizzes>
Exams:<bs><number of exams><bs> , <bs><weightage of exams>
<1 bl>
<column headers>
<a line>
<student data>
```

You can assume the following. Every class will have at least one homework and one exam. A class may or may not have a quiz. Each homework carries the same weight and so does each quiz (if there is one) and each exam. The weightage of homeworks, quizzes, and exams add up to 100.

Each student's data will occupy a separate line of input. The first column contains the student's name. The second column contains the student's social security number. The social security number will contain a hyphen after the third digit and the fifth digit. The third column contains information regarding the grade option for the students. The choices are “A-F”, “P/F” (Pass or Fail), and “AUD” (Audit). The fourth column either contains a ’Y’ or ’N’ depending on whether the student wants his/her grade to be posted publicly. Each subsequent column contains scores out of 100 for the hws, quizzes, and exams. The scores will be non-negative numbers. Some entries from column 3 onwards may be missing for students who audit the course.

A sample input file is given below.

```
Class :BIO:198:01
Semester :Fall 2004
Institute :The University of Adelaide
Instructor:Prof. Richie Benaud
Title:Final Grades

Homeworks:  3, 25
Quiz:  0, 0
Exams:  2, 75

 Name               SSN      Opt   Post   Hw1   Hw2   Hw3    Exam1   Exam 2
--------------------------------------------------------------------------
Seema Malhotra    099-66-7871  A-F    Y     100   67    81     59      98
Tim Crawford, Jr.  999-88-7766  A-F    N      99   100   96     91      90
Amit B. Badgamia 000-00-0110  AUD           5    100   28     19      76
Tim Shaw         398-72-3333  P/F    Y    76     84    49     69      78
Mehroo Wadia   909-37-3689  A-F    Y     97     94   100    61      79
```

Output Your program must query the user with the following message until the user decides to terminate the program by entering ’`Q`’ or ’`q`’: “`(S)ummarize, (F)ull Display, (R)ange, or (Q)uit:  `”.

If the user inputs ’`S`’ or ’`s`’, your program must output the information in the following format.

```
<course number>,<1 bs><year>
<university>
<instructor>
<title>
<1 bl>
<column headers>
<a line>
<student grades>
<1 bl>
Highest:<bs><highest total score>
Lowest:<bs><lowest total score>
```

```
Average:<bs><average score>
```

The column headers in the output must be "Id", "Hws", "Quizzes", "Exams", "Total", "Grade". If a student is not auditing a course and would like his/her scores to be posted then the student's grades will appear in the output. Each student's grades will occupy a separate line of output. The maximum score in each category would be the weightage of the corresponding category. Thus, if the hws were 25% of the total grade then the maximum score in the Hws column must be at most 25. All scores in the second, third, and fourth column must be precise up to two decimal places. If there is no quiz in the course then each entry in the column for Quizzes will be a -1.00. Each entry in the "Total" column must be an integer that is the *rounded* value of the sum of the homework score, quiz score (if there is a quiz), and exam score for that student. The "Grade" column must contain a letter grade for the student. The rule for giving a letter grade is as follows. A score of at least 90 corresponds to an 'A' grade, a score between 80 and 89 corresponds to a 'B' grade, a score between 70 and 79 corresponds to a 'C' grade, a score between 60 and 69 corresponds to a 'D' grade, and a score below 60 corresponds to an 'F' grade. A student who has chosen the "P/F" option gets a 'P' if the student has a total of at least 70, and gets an 'F' otherwise.

The statistics of interest are the highest score, the lowest score, and the average. All these statistics must take in to account the scores of all students except the ones who are auditing the course. The average score must be precise up to two decimal places. These scores must be right-aligned with the scores in the "Total" column.

For the above sample input, if the user requests a summary then the output must be as follows.

```
BIO:198:01, Fall 2004
The University of Adelaide
Prof. Richie Benaud
Final Grades

Id       Hws     Quizzes     Exams     Total       Grade
---------------------------------------------------------
7871    20.67    -1.00       58.88        80           B
3333    17.42    -1.00       55.13        73           P
3689    24.25    -1.00       52.50        77           C

Highest:                                  92
Lowest:                                   73
Average:                               80.50
```

If the user input is '`F`' or '`f`', then your program must also display the student's name in addtion to the other information. Information on all students not auditing the course must be displayed. Hence, for the sample input, the full display would be :

```
BIO:198:01, Fall 2004
The University of Adelaide
Prof. Richie Benaud
Final Grades

Name                Id      Hws     Quizzes     Exams     Total      Grade
---------------------------------------------------------------------------
Seema Malhotra      7871    20.67   -1.00       58.88       80          B
Tim Crawford, Jr.   7766    24.58   -1.00       67.88       92          A
Tim Shaw            3333    17.42   -1.00       55.13       73          P
Mehroo Wadia        3689    24.25   -1.00       52.50       77          C

Highest:                                                    92
Lowest:                                                     73
Average:                                                 80.50
```

You are free to choose the width of each column. However, the output must be well-spaced and all scores in a column must be right-aligned with each other. Also, you can output the student names in any order that you like.

If the user input is 'R' or 'r' then you must query the user for range using the following message: `Enter the range separated by ',':` . After the user enters the range (you may assume that both inputs are integers) your program must output the list of names of students who are not auditing and whose total score is within the range (both ends of the range inclusive). So for the sample input, if the user requests scores in the range 73 to 81, the output should be

```
3 student(s) have total score between 73 and 81.
           Tim Shaw
           Mehroo Wadia
           Seema Malhotra
```

Modularization A program that is not well modularized will be penalized heavily. A suggested way to modularize your program is to have the following functions. You can have additional functions if you wish.

A. `classInfo`
This function will process the first five lines of the input file that gives information about the course.

B. `gradingInfo`
This function should process the three lines in the input file that contain information about the numbers and weights of homeworks, quizzes, and exams.

C. `studentScores`
This function should process the remainder of the file. You may want to compute the letter grade of each student while processing the scores. This can be made as a nested function.

D. `stats`
This function should output information based on user input. You may want to have nested functions that that be invoked for different user inputs.

Note: For this program you must not use the `re` module and its methods.

---

[40] **2. Duplicates.** Write a program that checks if there are any duplicate names in a file. Specifically, your program must read into an array the names from the input file, "names.dat". Each name in the input file appears on a separate line and there will be at most 1000 names in the input file. Your program must have a function called `duplicates` that returns `true` if there are duplicate names in the file and returns `false` otherwise. This function must be implemented *recursively.* Your program must be case-sensitive.

For example, suppose the input file consists of names `Ron, Bob, Tim, Shinida, vijay, bob` then your program must output `There are NO duplicate names`. However, if the above input file also has another Tim, then your program must output `There are duplicate names.`

---

[40] **3. Course Selection.** It is that time of the semester when you have to register for courses for the next semester. After looking at the course offerings for next semester you find out that you like n courses. However, due to various constraints such as time, you want to register for exactly k courses. Write a program that would find out the number of different possibilities for choosing k courses out of n possible choices. The number of possibilities is given by $\frac{n!}{k!(n-k)!}$. You must not use this formula to implement your program, you can use it to verify your answer. Your program must query the user for n and k based on the sample runs given below. The user may enter any non-negative integers for n and k. Your program must have a function `choices` that takes in the two parameters n and k and returns the number of possibilities. The function `choices` must be implemented *recursively.*

Below are two sample runs of the program.

```
Number of courses you like:  5
Number of courses you can register for:  3
Total number of ways of choosing 3 out of 5 courses:  10

Number of courses you like:  10
Number of courses you can register for:  4
Total number of ways of choosing 4 out of 10 courses:  210
```

[*Hint:* Pick a course. Consider two possibilities. You either register for that course or you don't. In both the cases, the problem reduces to a smaller subproblem.]

Submission Guidelines. You must create a directory called P1_<last name> under which you must create three more directories for the problems. The directories must be named `gradeStats`, `duplicates`, and `courseSelection` and they must contain all files for the respective problems. Furthermore, this directory should not contain files that are not directly relevant to the problem. In each of the directories there must be a "README" file that must contain instructions for the user to run the program. The README file may also contain other instructions that would be helpful to the user of the program.

Before you submit your program, you must create a *compressed*, *tar* file of the directory P1_<last name>. This can be done in two steps as follows.

```
% tar cvf P1_<last name>.tar P1_<last name>
% gzip P1_<last name>.tar
```

The above commands should create a file P1_<last name>.tar.gz . You must e-mail the above file as an attachment to `rajivg@clam.rutgers.edu`. The programs are due by 11:59p.m. of the due date. Hard copies of your programs must be turned in by the end of the first class after the deadline.

Some other commands that may be useful are the following.
Uncompressing a file can be done using the `gunzip` command.

```
% gunzip <file name>.gz
```

Extracting files from an archive can be done as follows.

```
% tar xvf <file name>.tar
```

To list all the files in the archive do the following.

```
% tar tf <file name>.tar
```