Assignment 3 CSc 110
Grade Calculator

Motivation

In this assignment you are given the opportunity to experiment with conditional statements and nested loops and continue to develop your skills in deciding how to 'method-ize' a program. In addition, it introduces the use of an array. There are many choices to be made in the development of this code, different programmers will make different choices. Like two different works of art, there is more than one way to make a beautiful program. Taking an opportunity to explore alternative beautiful programs is an excellent learning experience.

This assignment represents an excellent opportunity to examine test strategy. The various inputs can all be tested for correctness. Try to consider all the possible incorrect inputs.

Learning Outcomes

- How to implement conditional statements (`if`-statements and `if/else`-statements)
- How to analyze the flow of choices to ensure correct program logic.
- How to use an array to store many pieces of data
- How to design a suite of tests that will check the various cases in a branching program (i.e. one that contains conditional statements).
- How to use parameterized loops within methods.
- How to design and use indefinite loops to handle invalid inputs.
- How to *refactor* your program design based on new constraints.
- How to model multiple, linear dimensions of a problem using nested loops: the nesting may occur across method boundaries.

Details

This assignment gives you the opportunity to design and implement a fairly substantial program! Be sure to apply the principles you have learned in the course so far---use constants, methods, and control structures in ways that create high-quality software. The problem requires the creation of an automated CSc 110 grade calculation program. Part I is interactive and prompts the user for all the scores for one student. The result is the percentage and letter grade for one student. Part II modifies Part I and makes the program read from an input file, and produces percentages and grades for all of the students in the class.

For the CSc 110 course, the course outline shows that grades are calculated as follows:

Assignments (Part 1)	(5 @ 1% each)	5%
Assignments (Part 2)	(5 @ 3% each)	15%
Lab Attendance	(10 @0.5% each)	5%
Quiz	(best of 2)	3%
Midterm		22%
Final Exam		50%

All assignments are marked out of a maximum of 100 marks each. The lab attendances will be added together and input as a single number (out of a maximum of 5). The two quizzes are marked out of a maximum of 20. The midterm and final are each marked out of a maximum of 100 marks.

Using the grades for each component and the weightings above a total course mark (maximum possible: 100), the final percentage, is converted to a letter grade according to the following scale:

F	D	C	C+	B-	B	B+	A-	A	A+
0-49.9	50-54.9	55-62.9	63-69.9	70-74.9	75-79.9	80-84.9	85-89.9	90-94.9	95-100

In order to pass the course, a student must obtain a passing grade in each of the following: (a) overall final mark; (b) final exam. Thus, anyone who receives a grade equal or greater than 50, but fails the final exam will receive an F.

Exactly what constitutes a 'passing grade' on the final exam and for the overall mark in the course is determined by the instructor at the end of the term. (The instructor may determine, for example, that the final exam was particularly difficult and decide that the passing grade on the final exam is 42%!!)

Planning Activity: Program Design Worksheet

Complete Program Design Worksheet (below), using the problem data as described above. Be sure to use proper pseudo-code that is **not** a program. You may discuss your completed worksheet with the instructor in office hours or with consultants.

Assignment 3 Part 1: Interactive Portion, Calculating a Grade for One Student

For this part of the assignment, create a grade calculator program that prompts the user for all of the grades for one student. (Specifically, the user will input each of the grades on the keyboard.) Each input should be a raw score out of the maximum possible for each component (a number out of 100 for each assignment, a number out of 20 for the quizzes, etc.). In the event a number exceeds the maximum or is less than zero, the user should be prompted to re-enter the value. The determination, by the instructor, of the minimum passing grade on the final exam must also be input.

The program will calculate and output the student's final percentage and letter grade. Sample output for this program is:

```
C O U R S E   G R A D E   C A L C U L A T O R
  Author: L. Jackson October 2010
  Purpose: Calculated the weighted grade for a student in a course
        Inputs: Assignment, Lab, Quiz and Exam grades

Passing Grade for Final ==> 50

Input ID number ==> 12345

Input Assignments (Part 1)(maximum 100) :
  #1==>80
  #2==>80
  #3==>80
  #4==>80
  #5==>80

Input Assignments (Part 2)(maximum 100) :
  #1==>80
  #2==>80
  #3==>80
  #4==>80
  #5==>80

Input Lab Attendance(maximum 5) :
  #1==>4

Input quizzes(maximum 20.0) :
  #1==>16
  #2==>16

Input Midterm(maximum 100) :
  #1==>80

Input Final Exam(maximum 100) :
  #1==>80

12345 Grade = 80.0 Letter = B+
```

Once this part of the assignment appears to be producing the correct output: run it many times with varying inputs (ie, test it carefully) and correct any errors that you find. Also, carefully evaluate your use of constants, methods, and control structures to ensure in ways that create high-quality software: please discussion your code with the consultants and/or the instructional team to find opportunities for improving the quality.

Submit your working Part 1 program before your lab during the week of October 25-29, 2010.

Assignment 3 Part 2: File-Based Portion, Calculating the Grades for the Entire Class!

Modify the previous program to read its input from a file called *scores.dat*. The first line of the file is an integer number, indicating the number of student records in the file. Before testing your program on a large file, test it on a file that contains only one 1 student record. Then make it work for a file that contains a small number of records, and finally test it with a larger file. The data for each student in the file will adhere to the following format, where each element is separated by either a space or a tab character and the lines are separated by new-line characters:

<number of records>
<student number> <a1p1> <a2p1> <a3p1> <a4p1> <a5p1> <a1p2> <a2p2> <a3p2> <a4p2> <a5p2> <lab> <quiz1> <quiz2> <midterm> <final>
<student number> <a1p1> <a2p1> <a3p1> <a4p1> <a5p1> <a1p2> <a2p2> <a3p2> <a4p2> <a5p2> <lab> <quiz1> <quiz2> <midterm> <final>
<student number> <a1p1> <a2p1> <a3p1> <a4p1> <a5p1> <a1p2> <a2p2> <a3p2> <a4p2> <a5p2> <lab> <quiz1> <quiz2> <midterm> <final>
.
.

It is important to notice that this file only contains the integer portion of the student ID numbers (ie, the V00 part is not included.)

The output, written to the screen, should be the last four digits of the student number, the final percentage, and the final letter grade. Additionally, there should be a summary of the number of students who received each grade (that is, the number of A+'s, A,'s, B+'s, B's, B-'s, C+'s, C's, D's and F's).

```
 C O U R S E  G R A D E  C A L C U L A T O R

INPUT OPTIONS: 1- Interactive (for one student)
               2- From File (for many students)
    Choose 1 or 2 ==>2

24403 Grade = 87.935 Letter = A-
33892 Grade = 78.52 Letter = B
4304 Grade = 85.63 Letter = A-
13605 Grade = 61.545 Letter = F

Grade  Number Receiving Grade
  A+     0
  A      0
  A-     2
  B+     0
  B      1
  B-     0
  C+     0
  C      0
  D      0
  F      1
```

Submit your working, documented Part 2 program (that follows the style guideline) before your lab during the week of November 1-5, 2010.

More to Discover

- Would you trust your computer program to decide on your CSc 110 grade? Would you trust a computer to decide if you have completed all the requirements necessary to be awarded a degree? What about tracking your bank account or making a medical diagnosis? Doctors that are good at making diagnoses often rely on a vast range of knowledge and experience. They are able to sort through the available data to find the relevant pieces of information and then compare that information to patterns that they have seen before. They can also make deductions and inferences to discover (or lead them to inquire about) extra information that may not be obvious. Many of the pattern recognition and inference procedures that doctors use can be expressed using a computer program. Such programs are examples of *artificial intelligence*. In fact, computers can often recognize more nuanced patterns and perform more complex inferences that a person can, simply because they have larger working memory. However, the one thing that computers lack is common sense. If an aged patient shuffles into a doctor's office with a cane, complaining of ankle pain, the doctor can infer fairly quickly that the pain is probably not due to an acute running injury—a computer on the other hand would have to ask, "Did you hurt yourself running?"
- Computers can be programmed to recognize patterns, but they can also be used to *find* new patterns. The current buzzword for this process is data mining. Companies such as Amazon employ programs to search through vast databases looking for statistical connections between one piece of information and another. Once they have identified relationships between various items, these patterns can then be exploited to produce customized advertisements such as, "You recently bought *x*. If you liked it, you may also be interested in *y*." Google provides similar functionality, producing customized ads alongside your search results or email messages (in GMail).
- Some decision-making programs can even be used as an agent that acts on your behalf. Basically, the program is configured to makes choices based on settings that you supply. It can then respond to a large number of queries for you, or it can take action quickly to respond to time-sensitive queries when you are not available. This is exactly how online auctions work at sites like eBay. You set your maximum bid and an "agent" program automatically submits minimum bids for you until your max is reached. The agent sends you a notice every time it successfully or unsuccessfully bids on your behalf.

Program Design Worksheet

Name:

Input(s):	
Output(s):	
Example:	

Problem Description	**(Describe a <u>subset</u> of the problem that will be the focus of the first stage of program development.)**
Pseudo-Code:	
	(Implement and Test the above pseudo-code. Use Numbers from the example for testing.)

Repeat: Problem Description (for a subset of the problem), Pseudo code and Implementation.