University of Victoria
Department of Computer Science
CSC 110 Fundamentals of Programming
ASSIGNMENT 3

DUE: Thursday, February 15, 2018 before 7:50 am – By submission on ConneX only

Output: MUST Match Sample Output – Exactly

Indentation: MUST be Exactly 4 spaces or 1 tab

Input: Only 1 Scanner object can be used.

How to hand in your work

Submit the file (called `Grades.java`) that completely answers part II (below) through the Assignment #4 link on the CSC 110 conneX site. Please make sure you follow all the required steps for submission (including the final confirmation of your submission: Never 'Save as Draft', but submit and re-submit whenever you create a better solution.) When you have received an email confirming your submission, then you know it is submitted, if not, wait a few moments, then re-check what you have done and re-submit.

In this assignment you are given the opportunity to experiment with conditional statements and loops and continue to develop your skills in deciding how to 'method-ize' a program. 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 also represents an opportunity to examine test strategy. The various inputs can all be tested for correctness. Try to consider all the possible incorrect inputs.

Learning Outcomes

When you have completed this assignment, you should be able to:

- implement conditional statements (if and if/else)
- analyze the flow of choices to ensure correct program logic.
- design a suite of tests that will check the various cases in a branching program (i.e. one that contains conditional statements).
- use parameterized loops within methods.
- design and use indefinite loops to handle invalid inputs.

Part I*: Problems from the Textbook*

- Complete the Chapter 3G Self-Check Problems questions 1 through 4 and Chapter 4 Self-Check Problems questions 1 through 10, 12, 14 through 17 and compare your answer to those given

by the textbook authors at: http://www.buildingjavaprograms.com/self-check-solutions-4ed.html

Part II: Coding Assignment Instructions

This assignment creates an automated CSc 110 grade calculator. The program is interactive and prompts the user for all the scores for one student. The result is the percentage and letter grade.
For the CSc 110 course, the course outline shows that grades are calculated as follows:

Coursework Weight (out of 100%)
Assignments: 7 worth 3% each → 21%
Labs: 10 worth 0.5% each → 5%
Midterm Exams: 2 worth 16% each → 32%
Final Exam: 42%

All assignments are marked out of a maximum of 10 marks each. The labs are marked out of a maximum of 1 mark each. The two midterms and the final exam 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	50-59	60-64	65-69	70-72	73-76	77-79	80-84	85-89	90-100

Since grades are typically calculated with real (as opposed to integer) numbers, assume that the fractional part of real numbers greater than and equal to .5 are rounded up to the next integer and those less than .5 are rounded down to the preceding integer.

In addition, to pass the course, a student must obtain a passing grade on the final exam. Thus, a grade equal or greater than 50, but with a failed final exam will be changed to 49%, yielding a letter grade of F.

Note: Exactly what constitutes a 'passing grade' on the final exam is determined by the instructor at the end of the term. Thus, the instructor may determine, for example, that the final exam was particularly difficult and decide that the passing grade on the final exam is 48%!!

Here are some specific directions for Part I of this assignment:

- The program that prompts the user to input each of the grades on the keyboard. Each input should be a raw score out of the maximum (as stated above). 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 then calculate and output the student's final percentage and letter grade.

Once this part of the assignment appears to be producing the correct output:

→ Use methods to organize your program and remove redundancy: Seek out repeated (or almost repeated) parts of your code. If there are areas that are nearly the same, except for strings or variable names, you have a clear sign that a method with parameters could be used to reduce

redundancy. FYI: Your instructor's solution contains 141 lines; including code, comments and blanks lines. This 'method-ization' is a vital and expected part of this program. (IMPORTANT NOTE: Only open 1 new Scanner object: If it is needed in methods, the Scanner reference should be passed as a parameter.)

→ Run it many times with varying inputs, test it thoroughly, 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, as described in the coding conventions document.

A sample output for this program is on the next page.

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

Passing Grade for Final ==> 47

Input Assignment (maximum 10):
  #1==>7
  #2==>8
  #3==>9
  #4==>7
  #5==>8
  #6==>12
     Incorrect input, try again:
  #6==>9
  #7==>10

Input Lab (maximum 1):
  #1==>1
  #2==>1
  #3==>1
  #4==>0
  #5==>1
  #6==>0
  #7==>1
  #8==>0
  #9==>1
  #10==>1

Input Midterm (maximum 100) :
  #1==>75
  #2==>67

Input Final Exam (maximum 100) :
  #1==>59

 Grade = 68.40   C+
```

Additional sample inputs provided in Assignment 4 Folder.

Marking

Your mark will be based on:

- [1 mark] Your code compiles
- [2 marks] Your program runs and produces exactly the same output as the sample output, for the same input. IMPORTANT: You can only open 1 new Scanner object: more than 1 may produce output, but will not be considered correct.
- [1 mark] Your program produced exactly the same error messages when incorrect values are input as the sample output.
- [4 marks] Your code follows the coding convention precisely, in particular:
  - indentation using exactly 1 tab stop or 4 spaces within each layer of braces,
  - appropriate choice of identifiers
  - documentation comments, including header comment, class comment, and method description comments.
  - clear, consistent and concise implementation comments.
- [2 marks] Your code uses parameterized methods and return values, where appropriate, to ensure there is little or no code redundancy.

More to Discover (Not necessary at all: but some might enjoy!)

- 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, who 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 also to *find* new patterns. This is known as data mining. App vendors and marketing forms, 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*." It is certainly a growing industry!