# COMP1006/1406 – Fall 2016

Submit a single file called `assignment1.zip` to cuLearn.
There are 30 marks possible marks.

In assignment specifications, text appearing `like this` represents things that a user types when running or using a program. Text that appears `like this` represents output from your program.

## 1: Final Grade Computation [10 marks]

Write a program that computes a final grade, as specified in the course outline using the four term marks: assignments, tutorials, midterma and final exam.

The four term marks will be provided to the program via command line arguments (values passed to the program when you run the program from the console window in DrJava) and will always be given in the same order: A, T, M, F.

➡ Starting with the skeleton program provided, `Grades.java`, add code to the `computeGrade` and `roundGrade` methods so that the program outputs to standard output (`System.out`) the correct final grade. Do not change the `main` method. Your program will output a single line with the computed final grade rounded to exactly one decimal place. The grades provided from the command line can be integers or decimal numbers (all are given as a percentage). To allow for possible bonus marks, valid percentages can range from 0.0 to 105.0. The `roundGrade` function will take a double and return a string that represents that number rounded to exactly one decimal place.

Add comments, whitespace, and indentation to the code so that the entire file (`Grades.java`), not just the functions you are completing, follows the style guide. Marks will be deducted if you do not follow the style guide.

**Examples:** From the console window in DrJava (or from a shell),

`java Grades 78.1 63.2 80.3 78` will output the single line `77.2`

`java Grades 50 105 50 50` will output the single line `55.5`

`java Grades 105 105 1 98` will output the single line `49.5`

**Testing:** Create and submit another program called `GradesTesting`, that tests your `Grades` program. For each test case, your program should display the input (grades), the expected output of your program and the actual output. For example, a single test case might look like

```
String[] input = {"78.1", "63.2", "80.3", "78" };
System.out.println("Test 0:  " + java.util.Arrays.toString(input));
System.out.println(" Expected answer is 77.2");
System.out.println(" Answer obtained is " + Grades.main(input));
```

Your testing program should also follow the style guide. In particular, be sure you describe (in comments) each test case (or group of test cases). The expected and actual output should line up nicely on the screen.

Mark breakdown: 3 marks for Style, 3 marks for Testing, 4 marks for Correctness

Include both your `Grades.java` and `GradesTesting.java` files in your submission `assignment1.zip` file.

## 2: Grade Conversion [10 marks]

➨ In the provided `Convert.java` file, complete the two methods: `convertToLetter` and `convertToGradePoint`. The interface (specification) of the methods are given below.

```
public static String convertToLetter(double grade)
 /* Purpose: converts a given numerical grade to a letter grade        *
  * Input   : a number                                                 *
  * output  : the letter grade (F, D-, D, ..., A+) corresponding to the   *
  *           input grade if the input is valid, "Invalid" otherwise   */

public static int convertToGradePoint(String letterGrade)
 /* Purpose: converts a given leter grade to its equivalent grade point   *
  * Input   : A valid letter grade in the range D- to A+ (no F's)      *
  * output  : The grade point corresponding to the input letter grade  *
  *           Use 0 for F and -1 for Invalid input                     *
  * Note    : you MUST use a switch/case for this method               */
```

The conversion table is given as follows

| Letter | Grade Range | Grade Point |
|---|---|---|
| A+ | [90, 100] | 12 |
| A | [85, 90) | 11 |
| A- | [80, 85) | 10 |
| B+ | [77, 80) | 9 |
| B | [73, 77) | 8 |
| B- | [70, 73) | 7 |
| C+ | [67, 70) | 6 |
| C | [63, 67) | 5 |
| C- | [60, 63) | 4 |
| D+ | [57, 60) | 3 |
| D | [53, 57) | 2 |
| D- | [50, 53) | 1 |
| F | [0, 50) | 0 |
| Invalid | other | -1 |

Note that the range $[a, b)$ means any number $x$ such that $a \leq x < b$. That is, square brackets mean inclusive and parentheses mean exclusive.

Mark breakdown: 2 marks for Style, 8 marks for Correctness

Put your `Convert.java` file in your `assignment1.zip` file.

**Examples:**

`Convert.convertToLetter(60.0)` $\xrightarrow{\text{returns}}$ `C-`

`Convert.convertToGradePoint("C-")` $\xrightarrow{\text{returns}}$ `4`

`Convert.convertToLetter(49.9)` $\xrightarrow{\text{returns}}$ `F`

`Convert.convertToGradePoint("F")` $\xrightarrow{\text{returns}}$ `0`

## 3: Longest Streak [10 marks]

➨ In the provided `LongestStreak.java` file, complete the following method:

```
public static int[] longestStreak(boolean[] values)
 /* Purpose : computes the length and locations of all the  maximal        *
  *           sequences of consecutive true occurrences in values          *
  * Inputs  : values is a non-null array of booleans with length at least 1  *
  * Outputs : outputs an integer array with one or more elements           *
  *              - first element is the length of a maximal sequence of    *
  *                consecutive trues in the input array values             *
  *              - the next elements are the indexes of the starting points  *
  *                (in the input array values) of each of the maximal      *
  *                sequences of consecutive trues                          *
  *           if there are no true values in the input array, output [0]   */
```

Do not change the method signature (use the provided skeleton java file). Changing the method modifiers, return type, name or input arguments will result in zero correctness marks for this problem.

Mark breakdown: 2 for style, 8 for correctness

Put your `LongestStreak.java` file in your `assignment1.zip` file.

Examples:

`boolean[] test_case = {true, false, true, false, true};`

`LongestStreak.longestStreak( test_case )` $\xrightarrow{\text{returns}}$ `[1,0,2,4]`

`test_case = new boolean[]{false, false, false, true, true};`

`LongestStreak.longestStreak( test_case )` $\xrightarrow{\text{returns}}$ `[2,3]`

## Submission Recap

A complete assignment will consist of a single file (`assignment1.zip`) with the following four (4) files included: `Grades.java`, `GradesTesting.java`, `Convert.java`, and `LongestStreak.java`.