COMP1006/1406 – Fall 2017

Submit a single file called `assignment1.zip` to cuLearn.
There are 40 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: 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 letter 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)` returns `C-`

`Convert.convertToGradePoint("C-")` returns `4`

`Convert.convertToLetter(49.9)` returns `F`

`Convert.convertToGradePoint("F")` returns `0`

2: Longest Streak [10 marks]

➡ In the provided `Problem2.java` file, complete the method called `longestStreak`. There should be no other methods or attributes in your class. The contract (specification) of the method is given below.

```
public static int longestStreak(boolean[] values)
 /* Purpose: computes the length of a longest streak of consecutive         *
  *          true occurrences in the input argument values                  *
  * Input  : values is a non-null array of booleans with length at least 1  *
  * output : outputs the maximal number of consecutive trues found in       *
  *          the input array                                                */
```

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 `Problem2.java` file in your `assignment1.zip` file.

Examples: Your `longestStreak` method should behave as follows:

```
boolean[] test_case = new boolean[]{false, true, true, false};
Problem2.longestStreak( test_case )
```
returns `2`

```
test_case = new boolean[]{false};
Problem2.longestStreak( test_case )
```
returns `0`

```
test_case = new boolean[]{true, false, true};
Problem2.longestStreak( test_case )
```
returns `1`

```
test_case = new boolean[]{false, true, true, true, false, true, true, true};
Problem2.longestStreak( test_case )
```
returns `3`

Note: Examples are provided as a guide to help you get your programs/methods working as specified. However, having your code match the examples is not a guarantee that your code will receive high correctness marks. You must test your code more extensively.

3: Longest Streak Again [10 marks]

➡ In the provided `Problem3.java` file, complete the method called `longestStreak`. There should be no other methods or attributes in your class. The contract (specification) of the method is given below.

```
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 (in increasing order)       *
  *           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 `Problem3.java` file in your `assignment1.zip` file.

Examples: Using the same examples as Problem 2, the output would be (in the same order) `[2,1]`, `[0]`, `[1,0,2]` and `[3,1,5]`. Other examples include

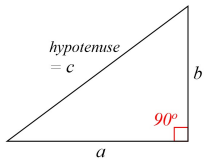
```
boolean[] test_case = {true, false, true, false, true};
Problem3.longestStreak( test_case ) --returns--> [1,0,2,4]

test_case = new boolean[]{false, false, false, true, true};
Problem3.longestStreak( test_case ) --returns--> [2,3]
```

4: Pythagorean Theorem [10 marks]

Recall Pythagoras's Theorem:

Pythagoras: In a right angle triangle, the square of the hypotenuse is equal to the sum of the squares of the two other sides.

If a, b and c are the lengths of the sides in a right angle triangle, with c being the hypotenuse (see diagram below), then $c^2 = a^2 + b^2$.



$$c^2 = a^2 + b^2$$

It is easy to check if three integer lengths (of a right angle triangle) satisfy this relationship. When the lengths are not integers though, we cannot just ask if $c^2 = a^2 + b^2$, because the numbers are no longer exact and checking if two approximate real numbers are equal to each other is difficult. Instead of checking for an exact equality, we

will say that the lengths satisfy Pythagorus' Theorem up to epsilon if, for some small epsilon $\epsilon > 0$, the absolute difference between c^2 and $a^2 + b^2$ is less than ϵ. That is,

$$\left|c^2 - (a^2 + b^2)\right| < \epsilon$$

The use of ϵ allows us to specify a tolerance for the equality check. That is, we can tolerate (or accept) an error in the check that is smaller than ϵ.

➡Write a Java class, called `Pythagorus`, that has three methods: `isPythagorusInt`, `isPythagorusDouble` and `main`. The first method will test if three integers satisfy Pythagorus' Theorem, the second will test if three approximate real numbers satisfy Pythagorus' Theorem (up to ϵ accuracy), and the last method will query a user for input to test numbers with the other two functions. The specifications for the first two methods are given below.

```
public static boolean isPythagorusInt(int a, int b, int c)
 /* Purpose : determine if a,b,c form a Pythagorean Triple                    *
  *           where c is the hypotenuse                                      *
  * Input   : a,b,c are each non-negative integers                           *
  * Output  : output true if c^2 = a^2 + b^2                                 *
  *           output false if not                                            */

public static boolean isPythagorusDouble(double a, double b, double c, double epsilon)
 /* Purpose : determine if a,b,c satisfy Pythagorus' Theorem up to epsilon    *
  *           where c is the hypotenuse                                      *
  * Input   : a,b,c are each non-negative doubles                            *
  *           epsilon is a non-negative doubles                              *
  * Output  : - outputs true if a,b,c satisfy Pyhtagorus' Theorem             *
  *             up to the given epsilon                                      *
  *           - outputs false if not                                         */
```

Note that this class must be a **program** so the `main` method must be public, static, return void have the right input argument type. The `main` method should repeatedly query the user for triples of numbers until they want to quit the program. First the user is asked if they want to enter integers, floating point numbers or quit the program. If they choose to enter numbers they are then asked to enter a triple of numbers (one input; three numbers comma separated). The program will outout `true` or `false` for the given triple. This is repeated until the user decides to quit with the String `"quit"`. Valid input for the first question are the strings `"int"`, `"float"` and `"quit"`. We will only test your code with these three strings (and various numbers).

Mark breakdown: style : 2 marks for style, 8 marks for correctness

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

Examples:

```
Enter int, float or quit : int
Enter triple : 3, 4, 5   displays→  true

Enter int, float or quit : float
Enter triple : 3.0, 4.0, 5.01, 0.0001   displays→  false

Enter int, float or quit : quit
good-bye
```

Submission Recap

A complete assignment will consist of a single file (`assignment1.zip`) with the following four (4) files included: `Convert.java`, `Problem2.java`, `Problem3.java`, and `Pythagorus.java`.