ASSIGNMENT 1

COMP-202, Fall 2015, All Sections

Due: October 5^{th}, 2015 (23:59)

Please read the entire pdf before starting.

You must do this assignment individually and, unless otherwise specified, you must follow all the general instructions and regulations for assignments. Graders have the discretion to deduct up to 15% of the value of this assignment for deviations from the general instructions and regulations. These regulations are posted on the course website. Be sure to read them before starting.

Question 1:	30 points
Question 2:	30 points
Question 3:	40 points
	100 points total

It is very important that you follow the directions as closely as possible. The directions, while perhaps tedious, are designed to make it as easy as possible for the TAs to mark the assignments by letting them run your assignment, in some cases through automated tests. While these tests will never be used to determine your entire grade, they speed up the process significantly, which allows the TAs to provide better feedback and not waste time on administrative details. Plus, if the TA is in a good mood while he or she is grading, then that increases the chance of them giving out partial marks. :)

Note that *none* of the graded questions on this assignment use the Scanner class. You should not use Scanner at any point in the graded questions. If you have any questions about this, please ask a TA or instructor.

Free pass usage

If you wish to use your "free pass" (see the course webpage for more details), you are required to email the TA who is marking your assignment *before* the deadline. Please see the table below, which is referenced by the *last* name of the student, to determine your TA. Note that this is also the person whom you should see if you have questions or other complains about your marks.

TA Name	Email Address	Student Last Name Range
Mohammad Patoary	mohammad.patoary@mail.mcgill.ca	A-BAN
Xiaozhong Chen	xiaozhong.chen@mail.mcgill.ca	BAO-BUDD
Stephanie Laflamme	stephanie.laflamme@mail.mcgill.ca	BUDE-CHR
Teng Long	teng.long@mail.mcgill.ca	CHS-DOW
Andrew Holliday	andrew.holliday@mail.mcgill.ca	DOX-GAG
Carlos Gonzalez Oliver	carlos.gonzalezoliver@mail.mcgill.ca	GAH - HARB
Hua Qun (Robin) Yan	huaqun.yan@mail.mcgill.ca	HARC - JON
Feras Abu Talib	ta_feras@hotmail.com	JOO - KOR
David Bourque	david.bourque@mail.mcgill.ca	KOS - LI, H
Paul Husek	paul.husek@mail.mcgill.ca	LI, I - MA
Babak Samari	babak@cim.mcgill.ca	MAC - MUR
Seyyed Mozafari	sh.mozafari@mail.mcgill.ca	MUS - PARE
Egor Katkov	egor.katkov@mail.mcgill.ca	PARF - RENT
Chenghui Zhou	chenghui.zhou@mail.mcgill.ca	RENU - SHIM
Jonathan Campbell	jonathan.campbell@mail.mcgill.ca	SHIN - TANG
Neeth Kunnath	neeth.kunnath@mail.mcgill.ca	TANN - WANG, Q
Ayush Jain	ayush.jain@mail.mcgill.ca@mail.mcgill.ca	WANG, T - YANI
Tzu-Yang (Ben) Yu	tzu-yang@mail.mcgill.ca	YAY- Z (end)

Part 1 (0 points): Warm-up

Do **NOT** *submit this part, as it will not be graded. However, doing these exercises might help you to do the second part of the assignment, which will be graded. If you have difficulties with the questions of Part 1, then we suggest that you consult the TAs during their office hours; they can help you and work with you through the warm-up questions. You are responsible for knowing all of the material in these questions.*

Warm-up Question 1 (0 points)

Create a file called `HelloWorld.java`, and in this file, declare a class called `HelloWorld`. This class should define only one method called `main()`. In the body of this method, use `System.out.println()` to display "`Hello world!`". You can find such a class in the lecture slides; make sure you can compile and run it properly.

Warm-up Question 2 (0 points)

Create a file called `A.java`, and in this file, declare a class called `A`. This class should define only one method called `main()`. In the body of this method, use `System.out.println()` to display the following pattern:

```
   A
  A A
 AAAAA
A     A
A     A
```

Warm-up Question 3 (0 points)

Practice with Binary:

Humans usually operate in base 10, probably because most of us have ten fingers. When operating in base 10, we think of numbers as having a `one`s column, a `ten`s column, a `100`s column, etc. These are all the powers of 10.

There is nothing special about 10 though. This can in fact be done with any number. In base 2, we have each column representing (from right to left) 1,2,4,8,16,etc. In base 3, it would be 1,3,9,27, etc.

Answer the following short questions about number representation and counting.

1. In base 10, what is the largest digit that you can put in each column? What about base 2? Base 3? Base n?
2. Represent the number seven in base 7.
3. Represent the number seven in base 2.
4. What binary number is equal to the sum of these two binary numbers? 11001101 + 100101010
5. What is the number from the previous part in base 10?
6. What is the binary number for 11010010 - 11000101?
7. And what is the number from the previous part in base 10?

Warm-up Question 4 (0 points)

This question is designed to help you practice with types.

Background: As discussed in the lectures, there are different types of values in programming languages such as Java. For instance, we can have integers like the number 5, real numbers like the number 5.1, boolean values like true and false, characters like 'm', or more complex types like Strings e.g. "we all live in a yellow submarine". We can compare, contrast, display and apply operators to values of these different types.

Recall also that we can assign values to variables that are declared with the appropriate type: char for storing characters, int for integers, double for real numbers, boolean for logical values, and several others.

When you declare a variable in Java, you need to specify the type. This involves thinking about a real world piece of data and figuring out what type you should use.

For each of the following, what type would best represent the data? Are there any that cannot be represented in Java?

1. The name of a day in the week.
2. Your letter grade in the course.
3. Your numeric grade in the course (as a percentage).
4. The name of the planet that your COMP 202 teacher is from.
5. The temperature outside.
6. The mathematical number π
7. The name of the author who wrote the Hitchhiker's Guide to the Galaxy?
8. The average number of brain cells in a human.
9. The average number of brain cells of a McGill student after three years of parties.
10. The (estimated) number of molecules in the universe.
11. The result of multiplying two integers together
12. The result of dividing an integer by another integer.
13. "Is the date today Sept 11?"
14. Does this string "i am happy" start with the character 'i'?

Warm-up Question 5 (0 points)

Logic

1. What does the following logical expression evaluate to?

```
(False or False) and (True and (not False))
```

2. Let a and b be boolean variables. Is it possible to set values for a and b to have the following expression evaluate as False?

```
a or (((not b) or (not b)) or (b or (not a)))
```

Warm-up Question 6 (0 points)

Write a method `swap` which takes as input two int values x and y. Your method should do 3 things:

1. Print the value of x and y
2. Swap the values of the variables x and y, so that whatever was in x is now in y and whatever was in y is now in x
3. Print the value of x and y again.

For example, if your method is called as follows: `swap(3,4)` the effect of calling your method should be the following printing

```
inside swap: x is:3 y is:4
inside swap: x is:4 y is:3
```

Now, create 2 int (integer) variables in the main method. Call them x and y. Assign values to them and call the swap method you wrote in the previous part.

After calling the `swap()` method—inside the main method— print the values of x and y. Are they different than before? Why or why not?

Warm-up Question 7 (0 points)

Consider the following 2-d matrix:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

Write a Java program that first reads 4 doubles, representing a,b,c, and d from the keyboard. It then outputs to the screen the determinant of the matrix.

For a 2x2 matrix, the determinant is always equal to $a \times d - b \times c$

Warm-up Question 8 (0 points)

Now consider the same question except on a 3x3 matrix:

$$\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$$

Write a Java program that first reads 9 doubles, representing the 9 letters above from the keyboard. It then outputs to the screen the determinant of the matrix.

For a 3x3 matrix, the determinant is always equal to

$$a \times (i \times e - f \times h) - b \times (d \times i - f \times g) + c \times (h \times d - e \times g)$$

Part 2

The questions in this part of the assignment will be graded.

Question 1: My first (graded) Java program (30 points)

The following should go inside a class called `MakingChange` and thus a file called `MakingChange.java`

Write a Java program that takes as input an integer number and outputs the minimum number of coins that can be used to make change. For the purposes of this question, assume that we have the following coin denominations available: 1 cent, 5 cents, 10 cents, 25 cents, 1 dollar (100 cents) and 2 dollars (200 cents). We have no bills.

The program should obtain its input from **String[] args** in the main method. **This is a different way to get input than Scanner. Do not use Scanner on this part.** That means that the input will initially be of type **String**. The input will be in the first position in a list of Strings. You can access this String-type list element and convert the it to an integer using the following command:

int money = Integer.parseInt(args[0]);

When making change, if we want to use the smallest possible number of coins, we will do this using something that in (advanced) computer science courses you will learn is a *greedy* algorithm. First, use the largest possible number of 2 dollar coins. Then, for the remaining amount, use the largest possible number of 1 dollar coins. Then, for the remaining amount, the largest possible number of 25 cent coins, etc.

For example, if we would like to make change for 1.46$, we would do the following:

- We can fit zero 2-dollar coins in 146 cents $146 - (0 \times 200) = 146$.
- We can fit one 1-dollar coin in 146 cents. $146 - (1 \times 100) = 46$.
- We can fit one 25-cent coin in 46 cents. $46 - (1 \times 25) = 21$.

- We can fit two 10-cent coins in 21 cents. $21 - (2 \times 10) = 1$
- We can fit zero 5-cent coins in 1 cent. $1 - (0 \times 5) = 1$
- We can fit one 1-cent coin into 1 cent. $1 - 1 \times 1 = 0$.

So given an input of 146, the program should output:
Change for 146 cents is:
Number of toonies: 0
Number of loonies: 1
Number of quarters: 1
Number of dimes: 2
Number of nickels: 0
Number of pennies: 1

You can assume that the input is valid: it is a non-negative integer number, where the last two digits are cents and any other digits denote dollars.

Question 2: Using Methods to Understand Leap Years (30 points)

The goal of this question is to have you write several methods that are all very similar in nature in order to experiment with the different sorts of methods. You should put all of your code into a class `LeapYearCalculator` *and thus a file* `LeapYearCalculator.java`*.*

Leap years occur ever 4 years, except when it's the turn of the century and the year is divisible by 100 (e.g. 2100). However if the year is divisible by 400, it's suddenly a leap year again. This is because the Earth's revolution around the sun is not precisely 365.25 days.

So 1900 is NOT a leap year (although 1900 is divisible by 4, it is also divisible by 100) but 2000 IS a leap year (although it is divisible by 100, it is also divisible by 400).

2a)Void method that prints something

Write a method `printIsLeapYear()` that takes as input an `int` argument representing a year and **prints** whether or not the year that is input is a leap year or not. You should include the year as part of the message. For example, your message could be "1994 is not a leap year" or "2000 is a leap year". Note that for full marks, this **must** be written on one line. (Remember that one way to do this is the `+` operator. Another way is to recall the difference between `System.out.print()` and `System.out.println()`).

Hint: To test your method, you can write a main method like you did in question one. The main method will not be graded, but without it, you won't know whether your method works or not! Your main method should call this method and verify the results. For example, your main method could be as follows. You should think of other cases to test!

```
public class LeapYearCalculator {
    public static void main(String[] args) {
        printIsLeapYear(1000);
        printIsLeapYear(4);
    }
    public static void printIsLeapYear(int year) {
        // your method definition here
    }
}
```

2b)Method that returns something

The previous method that you wrote is not very general. Remember that one of the key ideas of methods is to write a general piece of code that can be re-used.

What stops it from being general? Well, your method is only useful if you want to **print** whether something is a leap year. Suppose you wanted to check whether something was a leap year and take action based on that? For example, a calendar program may wish to check if it is currently a leap year, not for the purpose of printing it to the screen, but so that it knows whether to show 29 or 28 days in February.

We will now make another method *inside the same LeapYearCalculator class* that is more general.

Write a method `isLeapYear` inside of `LeapYearCalculator` that, like the previous method, takes as input an `int` representing the year to check, but returns a `boolean` value of `true` if it is a leap year and `false` if it is not a leap year. Your method should **not** print anything.

To test your method, you will need to call it from your main method. Think about how you can call the method and get it to display whether it is a leap year or not.

2c)A method calling another method!

Now that you have a more general leap year method, you can use it as part of a more complicated operation.

Write a method `subsequentLeapYear` that takes as input an `int` representing a year and returns an `int` representing the next leap year. If the input year is leap year, do **not** return the current year, but find the next leap year and return that one.

For example, if the input to the method is `1996` it should return `2000` since that is the next leap year.

Hint: You may choose to use a loop for this question, but it is not required. One way to do it is by a series of several if block. (In that case, think about what is the largest possible gap between leap years?) Another way to do it is by clever use of the modulus operator to determine the first multiple of 4 after the input number.

Hint 2: Remember that you can and should use your method isLeapYear in this section!

Question 3: DNA strand computations (40 points)

Background: Our DNA is double stranded consisting of the Watson strand and the Crick strand (after the two co-discoverers of the double-helical structure of DNA). Each strand can be viewed as a string over a four letter alphabet A, C, G, T. The Watson and Crick strands are paired up. The first letter of the Watson string matches with the first letter of the Crick string etc. Finally, the Watson and Crick strands are complementary: when the Watson strand contains an A, the Crick strand contains at T at that position (and vice versa); when the Watson strand contains a C, the Crick strand contains a G (and vice versa). This is called the Watson-Crick complement.

For example, if the Watson strand is "ACCAGACTAG", then the Crick strand will be "TGGTCTGATC".

When analyzing a DNA sequence, the DNA symbols are generally processed in groups of 3 letters. The Watson-Crick complement of a 3 letter sequence is calculated by taking each individual letter of the 3 letter sequence and calculating the complement of it. This new 3 letter String can then be used to figure out a corresponding Amino Acid. For more information on this process, see `https://en.wikipedia.org/wiki/Genetic_code`.

In this question, you should define a Java class called `DnaUtilities`. As a part of doing that, you will learn the very valuable skill of exploring the built-in Java libraries so that you do not need to "reinvent the wheel."

Question 3.1 Checking if something is a valid base.

Write a method `isValidBase` that accepts as input a value of type `char` and returns a value of type `boolean`. The method checks whether the value of the character is either A, C, G, T. So there are 4 possible char values that will return true.

Example invocation and output:

```
System.out.println(isValidBase('A'));//should print true
System.out.println(isValidBase('a')); //should print false
System.out.println(isValidBase('X') ); // should print false
```

For testing purposes, you should write a main method and call your method with various inputs. The main method will not be marked by TAs and so you can use it to test out your code thoroughly. The best way to make sure that your code works is to try really, really hard to break it.

Question 3.2 Finding the Watson Crick complement

Write a method `watsonCrickComplement` that accepts as input a value of type `char` and returns a char. The method should call `isValidBase` to make sure it is valid. If it is valid, it should find the Watson-Crick complement of the parameter (e.g. A becomes T, T becomes A, C becomes G, G becomes C), and return it as a char. If it is not valid, return the same character you started with.

1. `watsonCrickComplement('A')` returns 'T'
2. `watsonCrickComplement('t')` returns 't'
3. `watsonCrickComplement('4')` returns '4'

To test this question, you may add more code to the main method.

Question 3.3 Finding the Watson Crick complement of a 3 letter String

Write a method `watsonCrickTripletComplement` that accepts as input a value of type `String`, called *dnaSequence* and returns a `String`-type value.

The method should do the following things:

1. First, it should verify that `dnaSequence` has exactly 3 characters in it. (Hint: Use a method from the `String` class to accomplish this. See `http://docs.oracle.com/javase/7/docs/api/java/lang/String.html`) and examples below.
2. Next verify that each of the 3 characters are valid Watson-Crick complements. For this method, you should also use the documentation and the hint below.
3. If either of the above two steps fail, the method should return an empty String (e.g. `""`).
4. Finally, convert each of the 3 characters to their Watson-Crick complement, and return the `String`

Here are a few examples of the expected output:

1. `watsonCrickTripletComplement("AGA")` returns `TCT`
2. `watsonCrickTripletComplement("agA")` returns the empty `String`.
3. `watsonCrickTripletComplement("AAAA")` returns the empty `String`.

Hint: There are two methods that you may find very useful for this question. You are not required to use them, but we recommend doing so.

- `length()` : This method will tell you the number of characters in a String.
- `charAt(int i)` : This method will get the specific character at position `i` of the string, where i is an int. Note that the first letter is actually position 0. The 2nd letter is position 1, etc.

```
String message = "good-bye";
int length = message.length(); // stores 8, since 8 characters total.
char firstCharacter = message.charAt(0); // stores 'g', the first character.
char fifthCharacter = message.charAt(2 * 2); // stores '-', the 5th character.
char outOfRange = message.charAt(100); // results in a run time error, program crashes.
```

What To Submit

You have to submit one zip file with all your files in it to MyCourses under Assignment 1. If you do not know how to zip files, please ask any search engine or friends. Google might be your best friend with this, and a lot of different little problems as well.

These files should all be inside your zip.

`MakingChange.java`
`LeapYearCalculator.java`
`DnaUtilities.java`
`Confession.txt` (optional) In this file, you can tell the TA about any issues you ran into doing this assignment. If you point out an error that you know occurs in your problem, it may lead the TA to give you more partial credit. On the other hand, it also may lead the TA to notice something that otherwise he or she would not.