CSCI 1100 – 2017
Laboratory Report 5

Name:

Student ID:

Please indicate your registered lab room number:		
Rm 133	**Rm 134**	**Rm 143**

Declaration **Complete this section by filling in the shaded column to accurately reflect your work**		
1	This document is entirely my own work.	*Yes/no*
2	I obtained some help to complete this document.	*Yes/no.* *If yes, from whom? Give details.* *It is reasonable to obtain help from any person as long as you acknowledge the source.*
3	This document contains some material from the Internet or another document, file, or program.	*Yes/no.* *If you got help from a source you need acknowledge this (e.g., the URL or a book reference). You must detail how you used this source.* *Your lab work must still be your own. You should **not** copy and paste partial or full solutions.*

Your task is to complete this report using Word and JGrasp. You may use your own computer, or one of the lab computers provided.

Your submission should be a **PDF** copy of this document that includes your completed work as well as a **ZIP** file containing your source code files. You should submit both your **PDF document** and **ZIP file** on Brightspace: http://dal.brightspace.com

Submission Deadlines (firm):
Monday Labs due: Wednesday by 12:00pm (**noon**)
Friday Labs due: Sunday by 12:00pm (**noon**)

NB:

- Try to submit this report *during* the lab so that your TA can check it for you before you submit!
- Attendance is mandatory in all labs, and will form part of your overall lab grade
- Acknowledge any help that you obtained from friends, TAs, the Learning Centre, etc., by completing the Declaration on the first page of this document. Obtaining help is fine, *so long as you acknowledge it!*
- Any students who cannot log on to the lab computers should speak to the Help Desk to set up their account.
- Textbooks, class handouts, and any other materials are welcomed and encouraged in all labs!
- Food and drink are not permitted in the computer labs
- Late labs are ***not*** accepted! It is known that computer errors, power outages, and network lag are 105% more likely to occur between 11:55-11:59am, in the moment they can do the most damage. Account for this, and give yourself the chance to make a timely submission!

Exercise 0.
In the last Lab, you wrote a custom method for converting between two primitive data types. This functionality is already a part of the Java language, and is called *casting*. You may cast between different types of Objects (something we will see in CSCI 1101), or between certain primitive types. Answer the following questions about type casting, and remember to cite any sources you use.

a) Research the general syntax for casting between two primitive types in Java. Give the general syntax below.
Answers:
Using keyword "the general syntax for casting between two primitive types in Java" in Google.com, the first research result is: http://www.informit.com/articles/article.aspx?p=30871&seqNum=5
According to this article: The general syntax for convert a value in a type to another type is: **(typename)value**
From another article: http://way2java.com/casting-operations/data-type-casting-type-conversion/
There are two types of casting:
Implicit casting (widening conversion)
Explicit casting (narrowing conversion)
When a lower size data type is assigned to a higher size data type, the conversion will be done automatically.
Otherwise, a casting operation is a must: **(typename)value**

b) Give a short sample of code below that does the following. Declare an integer and a double variable. Initialize the integer to a value of your choice. Use casting to assign the same value to the double variable.
Answers:
```
int a;
double b;
a=10;
b=(double)10;
```

c) What is Java doing in memory when you cast a primitive type?
Answers:
From the article at: http://way2java.com/casting-operations/data-type-casting-type-conversion/
The memory size of different primitive type in java is different. For example, int type occupies 4 bytes, but double type occupies 8 bytes.
As a result, during a casting, the memory size will change accordingly. In implicit casting, the memory will increase, for example.

d) How does Java handle casting (in both directions) between char values and double values? To help answer this question, you may want to experiment with casting different double values and printing out the character result.
Answers:
According to this article: http://way2java.com/casting-operations/java-char-to-double/
The double type occupies 8 bytes and char type occupies 2 bytes. Generally, 2 bytes can be changed to 8 bytes value and this implicit casting will be done automatically by Java.
But when try to convert double to char, an explicit casting will be a must.

Exercise 1.
Write a method called `alternateNumbers` that takes a single integer parameter *n* and uses a loop to print off the integers between 1 and *n* on the same line, but where every second integer is negative. The method should also make clear what the parameter was when it was called. For instance, if you called your method like this

```
alternateNumbers(6)
```

then your method should print out something like this:

```
(6): 1 -2 3 -4 5 -6
```

Write an accompanying main method that calls `alternateNumbers` twice, once with the parameter 6, and then with a different positive integer parameter of your choosing. Include both methods in a class named `Exercise1` for submission, and paste your program's output below.

For this question and all following questions:

- remember to use comments in your code where appropriate!
- include your source code in a class named after the exercise number, like you did above

Solution (program output):

```
(6):  1 -2 3 -4 5
(7):  1 -2 3 -4 5 -6
```

Exercise 2.

The Fibonacci sequence are numbers in an integer sequence, where the first two values are 0 and 1, and each subsequent value is the sum of the previous two. Even though the sequence is infinite, you only need the previous two values to compute the next one! The third Fibonacci number is 1, the fourth Fibonacci number is 2, and so on.

Answer each of the following questions. Indicate your written answers and program output below, where appropriate.

a) The numbers 34, 55 form part of the Fibonacci sequence. What is the next value?

b) The numbers 144, 233, 377, 610 form part of the Fibonacci sequence. What is the next value? Identify which values you used in your calculation.

c) What is the next Fibonacci number after the one you computed in part (b)? Identify which values you used in your calculation.

d) Imagine you are locked in a room with a friend and three small (magical) whiteboards – two for you, and one for your friend. It looks like the only way to get out is to compute Fibonacci numbers! Sadly, your friend knows nothing about the Fibonacci sequence; however, they are a computational genius and can add together any two numbers instantly!

   Since these are magical whiteboards, you know that only *one* number can be written on each whiteboard, but the number may be of any size. In order to be released, you will be given two Fibonacci numbers, and your task will be to compute the next value in the sequence. Neither you nor your friend are allowed to speak or move, but they already understand that whatever two values you show them on your whiteboards, they should add them together and write it on their whiteboard to show you.
   Can you use the whiteboards to get your friend to compute the next Fibonacci number?
   Can you use the whiteboards to compute the next *two* Fibonacci numbers? The next *three*?
   Is there any limit to how many Fibonacci numbers you can compute this way?

e) Write a method called `fib` that takes an integer *n* as a parameter, and prints out the first *n* Fibonacci numbers.

f) Write an accompanying main method that calls `fib` twice, once with the parameter "15" and once with another positive integer parameter of your choice. Include both methods in the same class for submission, and paste your program's output below.

Solution (written answers):

a) Answers :
The next value will be 34+55=89

b) Answers:
Only the two previous value are necessary for calculating the next value, so next value will be 377+610=987

c) Answers:
The next value will be: 610+987=1597

d) Answers:
d.1 YES, because in order to calculate the next Fibonacci number, only the two previous numbers are a must.
d.2 YES, because I will have two whiteboards in which there should only one number can be written.
First step, I will write these two numbers in my whiteboards and show to my friend.
He/she see these two numbers, adding them and show me the result.
I will erase the first number and replace with the result, then show them to my friend.
So he/she can make a new calculation.
d.3 NO, on the above analysis, there is no limitation about calculating the Fibonacci numbers.

Solution (program output):

```
the first 15 Fibonacci numbers:0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
the first 16 Fibonacci numbers:0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610
```

Exercise 3.
Write a program that will print out all the prime numbers in a range of values, all on separate lines. Implement and use each method described below. Test each method on its own to make sure it behaves as you expect before moving on to the next. You may use any method in the implementation of any other subsequent method if you wish.

NB: Recall that a prime number is a positive integer that has no divisors other than 1 and itself. A divisor is any positive integer that "divides evenly" into the number. So for example, 11 is prime because there are no numbers x,y that will give x*y = 11 other than the divisors 1 and 11. The number 15 is not prime because 3*5 = 15, so both 3 and 5 are divisors. The number 12 is not prime because 2*6 = 12 or 3*4 = 12.

a) Write a method called `isADivisor` that takes two integer parameters. It returns true if the second parameter is a divisor of the first, and false otherwise. For example

```
isADivisor(18, 9)
```

would return true (because 9 is a divisor of 18), while

```
isADivisor(21, 4)
```

would return false (because 4 is not a divisor of 21).

Consider what value

```
isADivisor(11, 11)
```
should return. This is an example of an *edge case.* Be wary of edge cases later! Make sure they behave as you want them to.

b) Write a method called `isPrime` that takes a single integer parameter and returns true if the parameter has *no* divisors other than 1 and itself, and returns false otherwise. For example
```
isPrime(11)
```
would return true, while
```
isPrime(25)
```
would return false.

c) Write a method called `printPrimes` that takes two integer parameters, has no return value, and prints out on separate lines all the prime numbers that are greater than (or equal to) the first parameter and less than (or equal to) the second parameter. For example,
```
printPrimes(5,15)
```
would print out
```
5
7
11
13
```

d) Write the main method, and use it to print out all prime numbers between 200 and 400, all prime numbers between 2000 and 2200, and all prime numbers between 200000000 (two hundred million) and 200000200 (two hundred million and two hundred). You do not need to include your program's output below.

e) What differences do you observe when running your program and printing the values over these three separate ranges of 200 numbers each? Why is the behaviour different?

Solution (written answer):

There are 32 prime numbers between 200 and 400. For example, the first is 211, the last one is 397.
There are 24 prime numbers between 2000 and 2200. For example, the first one is 2003, the last one is 2179.
There are 14 prime numbers between 200000000 and 200000200. For example, the first one is 200000033, the last one is 200000183.

The reason of this different behavior is due to the algorithm about prime numbers. For example, the number 211 is a prime number, but the 2211 is not.

In order to check why 2211is not a prime number, write a method called printNumber that takes one integer parameter, has no return value, and prints out the value with which parameter has divisors other than 1 and itself. Details please finding in java code file in Exercise3.
The output of printNumber(2211) is as below:
```
3
11
33
67
201
737
```

Before submitting, check that:

1. You have properly filled in your name, ID, and Declaration on the first page,
2. You have included your solution for each question that requires one,
3. Your solutions are easy to read and formatted appropriately,
4. You have rick-rolled each of your lab TAs (optional),
5. You have saved your submission (which should be a completed copy of **this file**) as a **PDF**,
6. You have included *all* of your source files (one per question) in a **ZIP** file,
7. You are preparing to submit **both** your **PDF** and **ZIP** file on Brightspace, and
8. You have logged off of any lab computers.