First Name: _____ **Last Name**: _____

McGill ID: _____ **Section**: _____

Faculty of Science
COMP-202A - Introduction to Computing I (Fall 2010) - All Sections
Midterm Examination

Tuesday, November 2, 2010
18:30–20:30

Examiners: Maja Frydrychowicz [Section 1]
Mathieu Petitpas [Sections 2 and 3]

Instructions:

- **DO NOT TURN THIS PAGE UNTIL INSTRUCTED**
- This is a **closed book** midterm examination; notes, slides, textbooks, and other forms of documentation are **not** allowed. However, a letter-sized (8.5" by 11") **crib sheet** is permitted. This crib sheet can be single or double-sided, it can also be handwritten or typed, but the use of magnifying glasses is prohibited. Translation dictionaries (for human languages only) are permitted, but instructors and invigilators reserve the right to inspect them at any time during the examination.
- **Non-programmable calculators** are allowed (though you should not need one).
- **Computers, PDAs, cell phones, and other electronic devices** are **not** allowed.
- Answer **all** questions **on this examination paper** and return it. If you need additional space, use page 16 or the booklets supplied upon request and clearly indicate where each question is continued. **In order to receive full marks for a question, you must show all work** unless otherwise stated.
- This midterm examination has **18** pages including this cover page, and is printed on both sides of the paper. Pages 17-18 contain information about useful classes and methods.

1	2	3	Subtotal
/11	/6	/3	/20

4	5	Subtotal
/24	/6	/30

6.1	6.2	6.3	Subtotal
/15	/15	/20	/50

Total
/100

Section 1 - Short Questions

[11] 1. For each expression below, describe the order of evaluation taught in class by filling in the tables provided. For example, if you thought that the + operator is evaluated first, and the / operator is evaluated second in the expression `(3 + 5) / 2`, you would fill out a table as follows:

Operator	Context	Result
+	`3 + 5`	8
/	`(3 + 5) / 2`	4

The *Context* column should make it clear which part of the expression you are referring to. The last value in the *Result* column **MUST** be the overall value of the entire given expression.

Remember, explicit casting is also considered to be an operator.

a) `1 - 3 * (1 + 6) / 2`

Operator	Context	Result

b) `1 / 2 + 8 * (1 % 5 - 1)`

Operator	Context	Result

c) `2.5 * 2 + (int) 0.9 / 0.2`

Operator	Context	Result

d) `!(1 < 1) != (7 % 2 == 0) || 4 / 4 / 4 > 1`

Operator	Context	Result

[6] 2. What will be displayed to the screen after the following program is executed?

```
public class WhatWillBeDisplayed {
  public static void main(String[] args) {
    String s1, s2, s3;
    char a = '\n';
    s1 = "==";
    s2 = "*";
    s3 = s2;
    s2 = s1;
    System.out.println(1 + 2 + s3 + 4 + 5 + a + 4 * 5);

    int x = 9;
    double j = (int) 12.9;
    while(x+1 <= 11){
      j = j / 4;
      x++;
    }
    System.out.println(j);
  }
}
```

WRITE THE PROGRAM'S OUTPUT IN THE SPACE BELOW:

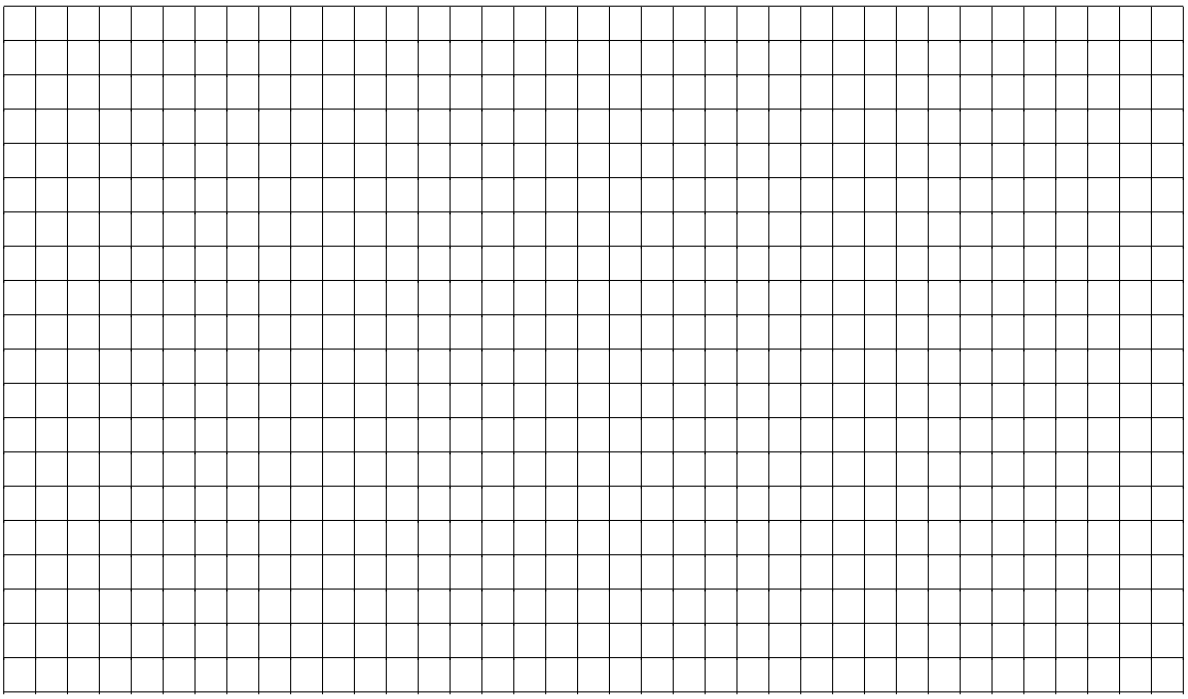
[3] 3. The following program prints a shape to the screen using the characters ⋆ and @.

Draw the program's exact output in the grid below. Each square in the grid can hold exactly one symbol or space. You **MUST** assume that the program prints its output starting at the top left corner.

```
public class Exercise {
  public static void main(String[] args) {
    final int LINES = 5;

    for (int y = 1; y <= LINES; y++) {
      for (int z = 1; z <= LINES - y; z++)
        // The String being printed below contains 3 spaces
        System.out.print("   ");
      for (int w = y; w >= 1; w--)
          // The String being printed below contains 2 spaces
          System.out.print("  " + '*');
      for (int w = 2; w <= y; w++)
          // The String being printed below contains 2 spaces
          System.out.print("  " + '@');

      System.out.println();
    }
  }
}
```



Total marks for Section 1: **20**

Section 2 - Long Questions

[24] 4. Consider the following program:

```
1   public class Errors
2    public static void Main(String[] args) {
3
4
5      for (int p = 2; p <= 15; p--) {
6        int i = (int) (Math.pow(2, p) - 1);
7
8        if (isPrime(i))
9          {System.out.println(p + "\t"
10                              + i + " is prime!");
11       else System.out.println("Not prime. Reversal\t"
12             + reversal(i));}
13     }
14   }
15
16   public static boolean isPrime(int num) {
17     if ((num == 1) || (num == 2)) {
18       return true;
19     }
20
21     for (int i = 2; i <= num / 2; i++) {
22       if (num % i = 0) {
23         return false;
24       }
25     }
26     return true;
27   }
28
29   public static int reversal(int num) {
30     final int result = 0;
31
32     while (num != 0) {
33       {int lastDigit = num % 10;}
34       result = result * 10 + lastDigit;
35       num = num / 10;
36     }
37   }
38 }
```

The above program is designed to check whether numbers x of the form $2^p - 1$ are prime for integer values of p between 2 and 15, inclusive. The program should display the result for each x. If a particular value of x is not prime, the program should also display this value of x in reverse. For example, 15 is not prime, so the program should print 51 along with a message about 15 not being prime.

However, there are **8** errors in this program. Find all the errors and list them. For each error you list, you **MUST** provide the **line number** at which the error occurs, the **type** of error (compile-time, run-time, logical), and a **description** of the error.

Do not list more than 8 errors, as you will be penalized for every "error" in excess of 8.

LIST THE ERRORS YOU FIND IN THE PROGRAM HERE:

[6] 5. Consider the following program:

```
1   public class Mystery {
2     public static void main(String[] args) {
3       int[] result = {4, 3, 44, 9, 5};
4       int i;
5       doSomething(result);
6       for(i = result.length - 1; i >= 0; i--)
7         System.out.println(result[i]);
8         System.out.println(i);
9     }
10
11    public static int[] doSomething(int[] input) {
12      int[] temp = new int[input.length];
13      int[] result = input;
14      temp[3] = 0;
15      input = temp;
16      input[0] = result[result[1]];
17      result = input;
18      if(input[0] < result.length-1)
19        result[0] = input[0] - 1;
20      return result;
21    }
22  }
```

Answer the following **four** questions.

a) What is stored in the variable `result` after line 3 is executed?

b) What is stored at `input[0]` and `input[1]` after line 12 is executed?

c) What is stored at `input[0]` and `input[1]` after line 16 is executed?

d) What is displayed on the screen when this program is executed?

Section 3 - Programming Questions

6. **Introduction**: In this three-part programming question, you will write a program that analyzes how popular people are on the website LikeBook, a fictional social network that keeps track of who "likes" whom.

   The LikeBook developers represent the "likes" relationship among a group of n people with an $n \times n$ table of boolean values, as shown in the following example for 3 people:

	0	1	2
0	`false`	`true`	`true`
1	`true`	`false`	`true`
2	`true`	`false`	`false`

   The value `true` corresponds to "likes", and the value `false` corresponds to "does not like". The second row of the above table is interpreted as follows, from left to right:

   - Person 1 likes Person 0
   - Person 1 does not like Person 1
   - Person 1 likes Person 2

   The second column is interpreted as follows, from top to bottom:

   - Person 0 likes Person 1
   - Person 1 does not like Person 1
   - Person 2 does not like Person 1

   In other words, row j of the above table expresses who Person j likes, and column j expresses who likes Person j.

[15] **Part 1**:

Write a class called `LikeBookUtils` that contains a method named `countValues()`. This method takes as parameters a 1D array of `boolean`s called `boolArray`, and a `boolean` called `target`. It returns an integer that represents the number of values in `boolArray` that have the same value as `target`.

You **MAY** assume that `boolArray` is not a `null` reference.

WRITE YOUR `LikeBookUtils` CLASS IN THE SPACE BELOW:

YOUR `LikeBookUtils` CLASS CONTINUED:

[15] **Part 2**:

Declare a second method in the `LikeBookUtils` class, called `computeMostPopular()`. This method takes as its sole parameter a 2D `boolean` array called `likeTable`, which corresponds to the "likes" relationships table described in the introduction, and returns the integer ID of the person who is liked by the most people.

If several people are liked by the same number of people, the method returns the lowest ID among those who are tied for most popular. In the example shown in the introduction, Person 0 and Person 2 are each liked by two people. In this case, the `computeMostPopular()` method should return the integer `0`.

You **MUST** assume that the array `likeTable` is organized in such a way that the first dimension represents the columns in our example table above, and the second dimension represents the rows. That is, if `likeTable` represents the table in our example above, the value stored in `likeTable[1][2]` is `false` and the value stored in `likeTable[2][1]` is `true`.

You **MUST** call the `countValues()` method you were asked to write in Part 1. You **MAY** assume that the `countValues()` method has been implemented correctly even if you did not successfully complete Part 1.

The declaration of the `computeMostPopular()` method is sufficient for this question; it is **NOT** necessary to place this declaration in the body of a class in order to get full marks for this question.

WRITE YOUR `computeMostPopular()` METHOD IN THE SPACE BELOW:

YOUR `computeMostPopular()` METHOD CONTINUED:

[20] **Part 3**:

Write a class called `PopularityAnalyzer` that contains one method called `main()`. This method should ask the user to enter the number of people for which to construct a "likes" relationship table, and then ask the user to enter whether a person likes another for every pair of distinct persons. Finally, using the collected information, the method computes the most popular person in the "likes" relationships table and displays the ID of this person to the screen.

Sample session:

```
Enter the number of people: 3
Person 1 likes Person 0? true
Person 2 likes Person 0? false
Person 0 likes Person 1? true
Person 2 likes Person 1? true
Person 0 likes Person 2? false
Person 1 likes Person 2? true
The most popular person is: 1
```

Note that Person i always dislikes themselves and therefore you program **MUST NOT** ask the user to enter whether Person i likes themselves.

You **MUST** call the `computeMostPopular()` method you were asked to write in Part 2. You **MAY** assume that the `computeMostPopular()` method has been implemented correctly even if you did not successfully complete Part 2.

WRITE YOUR `PopularityAnalyzer` CLASS IN THE SPACE BELOW:

YOUR `PopularityAnalyzer()` CLASS CONTINUED:

Total marks for Section 3: **50**

Total marks: **100**

USE THIS PAGE IF YOU NEED ADDITIONAL SPACE. CLEARLY INDICATE WHICH QUESTION(S) YOU ARE ANSWERING HERE.

SUMMARY OF JAVA STANDARD LIBRARY METHODS FOR SELECTED CLASSES

- `String` (package `java.lang`) Methods:
  - `public boolean equals(Object anObject)`: Compares this `String` to `anObject`.
  - `public boolean equalsIgnoreCase(String anotherString)`: Compares, ignoring case considerations, this `String` to `anotherString`.
  - `public int compareTo(String anotherString)`: Compares this `String` to `anotherString` lexicographically; returns a negative value if this `String` occurs before `anotherString`, a positive value if this `String` occurs after `anotherString`, and `0` if both `String`s are equal.
  - `public int compareToIgnoreCase(String anotherString)`: Compares, ignoring case considerations, this `String` to `anotherString` lexicographically; returns a negative value if this `String` occurs before `anotherString`, a positive value if this `String` occurs after `anotherString`, and `0` if both `String`s are equal.
  - `public char[] toCharArray()`: Converts this `String` to a new character array.
- `Scanner` (package `java.util`) Methods:
  - `public Scanner(InputStream source)`: Constructs a new `Scanner` that produces values scanned from the specified input stream.
  - `public double nextDouble()`: Scans the next token of the input as a `double`.
  - `public boolean nextBoolean()`: Scans the next token of the input as a `boolean`.
  - `public int nextInt()`: Scans the next token of the input as an `int`.
  - `public String nextLine()`: Advances this `Scanner` past the current line and returns the input read.
  - `public int nextLong()`: Scans the next token of the input as an `long`.
- `PrintStream` (package `java.io`) Methods:
  - `public void print(boolean b)`: Prints `boolean` value `b`.
  - `public void print(char c)`: Prints `char` value `c`.
  - `public void print(char[] s)`: Prints the array of `char` `s`.
  - `public void print(double d)`: Prints `double` value `d`.
  - `public void print(int i)`: Prints `int` value `i`.
  - `public void print(Object o)`: Prints `Object` `o`.
  - `public void print(String s)`: Prints `String` `s`.
  - `public void println()`: Terminates the current line by writing the line separator string.
  - `public void println(boolean b)`: Prints `boolean` value `b` and then terminates the line.
  - `public void println(char c)`: Prints `char` value `c` and then terminates the line.
  - `public void println(char[] s)`: Prints array of `char` `s` and then terminates the line.
  - `public void println(double d)`: Prints `double` value `d` and then terminates the line.
  - `public void println(int i)`: Prints `int` value `i` and then terminates the line.
  - `public void println(Object o)`: Prints `Object` `o` and then terminates the line.
  - `public void println(String s)`: Prints `String` `s` and then terminates the line.
- `Math` (package `java.lang`) Methods:
  - `public static double pow(double a, double b)`: Returns the value of `a` raised to the power of `b`.
  - `public static double sqrt(double a)`: Returns the correctly rounded positive square root of `double` value `a`.
  - `public static double random()`: Returns a `double` value with a positive sign, greater than or equal to `0.0` and less than `1.0`.
  - `public static double sin(double a)`: Returns the trigonometric sine of angle `a`, where `a` is in radians.
  - `public static double cos(double a)`: Returns the trigonometric cosine of angle `a`, where `a` is in radians.
  - `public static double tan(double a)`: Returns the trigonometric tangent of angle `a`, where `a` is in radians.
  - `public static double toDegrees(double angrad)`: Converts angle `angrad` measured in radians to an approximately equivalent angle measured in degrees.

  - `public static double toRadians(double angdeg)`: Converts angle `angdeg` measured in degrees to an approximately equivalent angle measured in radians.
  - `public static double exp(double a)`: Returns Euler's number e raised to the power of `double` value `a`.
  - `public static double log(double a)`: Returns the natural logarithm (base e) of `double` value `a`.
  - `public static double log10(double a)`: Returns the base 10 logarithm of `double` value `a`.

- `Character` (package `java.lang`) Methods:

  - `public static boolean isDigit(char ch)`: Determines if character `ch` is a digit.
  - `public static int digit(char ch, int radix)`: Returns the numeric value of character `ch` in the radix `radix`, `-1` if `ch` does not represent a digit in this radix.
  - `public static char forDigit(int digit, int radix)`: Returns the character representation of `digit` in the radix `radix`.
  - `public static boolean isLetter(char ch)`: Determines if character `ch` is a letter.
  - `public static boolean isLowerCase(char ch)`: Determines if character `ch` is a lowercase character.
  - `public static boolean isUpperCase(char ch)`: Determines if character `ch` is an uppercase character.
  - `public static boolean isWhitespace(char ch)`: Determines if character `ch` is white space according to Java.
  - `public static char toLowerCase(char ch)`: Converts character `ch` to lowercase.
  - `public static char toUpperCase(char ch)`: Converts character `ch` to uppercase.