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

McGill ID: _____ **Section**: _____

Faculty of Science
COMP-202A - Foundations of Computing (Fall 2016) - All Sections
Midterm Examination

October 31st, 2016
18:00 - 21:00

Examiners: Melanie Lyman-Abramovitch [Section 1 (MWF 8:35-9:25)]
Kaleem Siddiqi [Section 2 TR (8:35-9:55)]
Dan Pomerantz [Section 3 MWF (16:35-17:25)]

Instructions:

- **DO NOT TURN THIS PAGE UNTIL INSTRUCTED**
- This is a **closed book** examination; only a legal-sized (8.5" by 14") **crib sheet** is permitted. This crib sheet can be single or double-sided; it can be handwritten or typed. Non-electronic translation dictionaries are permitted, but instructors and invigilators reserve the right to inspect them at any time during the examination.
- Besides the above, only writing implements (pens, pencils, erasers, pencil sharpeners, etc.) are allowed. The possession of any other tools or devices is prohibited.
- Answer **all** questions on the answer sheet.
- This examination has **7** pages including this cover page, and is printed on both sides of the paper. On page 7, you will find information about **useful classes and methods**. **You may detach the Appendix (page 7 onwards) from the examination if you wish.**

Scoring

The exam will be scored as follows:

1. Questions 1 to 8 are worth 1 point each
2. Questions 9 to 23 are worth 2 points each
3. Question 24 is worth 32 points
4. Total: 70 points

Version-0

True/False Section (1 point each)

1. If you only expect numbers as command line arguments, you can change
   `public static void main(String[] args)` to
   `public static void main(double[] args)` and run the program.
2. The following is a valid header for a `main` method. (Valid → will compile and run).

   ```
   public static void main(String[] commandArguments)
   ```

3. You can put an if-block inside of another if-block.
4. A method can contain more than one return statement.
5. When iterating through the elements of array via its index, you must use a *for* loop as opposed to a *while* loop.
6. When you execute a class in Java you must *always* provide command line arguments.
7. What is the value of the boolean variable b?

   ```
   boolean b = (true||false||false)&&!(!true);
   ```

8. If x and y are positive integers, and $x < y$, then the result of $x\%y$ is always x.

Short Answer & Multiple Choice (2 points each)

For all of the 'what prints' questions in this section, if you believe that nothing will print as a result of executing the sample code, write 'N/A'. We are *not* evaluating the difference between `System.out.print` and `System.out.println`

9. What is the value of x?

   ```
   double x = (double) (1/2) + (1/2) + (double)(1/2);
   ```

10. The following method contains five variables. Which variables (if any) exist on the line marked ****HERE****?

```
public static void method(String[] input){
  int x = Integer.parseInt(input[0]);
  int y = Integer.parseInt(input[1]);
  if(x<y){
    String s = "The first number is smaller";
  }
  else {
    String t = "The second number is smaller";
  }
  ****HERE****
}
```

11. What prints when the following code executes?

```
public static void main(String[] args){
  int x = 4;
  int y = 5;
  y = swap(x,y);
  System.out.println("x is " + x + " y is " + y);
}

public static int swap(int x, int y){
  int temp = x;
  x = y;
  y = temp;
  return x;
}
```

12. What prints when the following code executes?

```
public static void main(String[] args){
  int x = 4;
  int y = 5;
  System.out.println(x + "+" + y + " is " + x + y);
}
```

13. What prints when the following method executes?

```
public static void method(){
  int n = 5;
  if(n>0){
    System.out.print("one ");
    n=0;
  }
  if(n<=0){
    System.out.print("two ");
  }
  else {
    System.out.print("three ");
  }
}
```

14. Consider the following method header:

```
public static void method(double x, int y)
```

Which of the following are *valid* ways to use this method? Write the *letters* corresponding to the correct answer(s) on your answer sheet. Note that there are five options total. Two are on the next page.

(a) `System.out.println(method(4,4));`

(b)
```
String s = "apple"
method(Math.pow(2,3), s.length());
```

(c)
```
double x=1.2;
int y=7;
method(double x, int y);
```

(d) `method(2.5, Math.sqrt(4));`

(e) `method(4, 4);`

15. You are writing a method called `findMax` that takes as input three decimal numbers and prints the value of the largest one. What does the *method header* look like? Just write the header. Do not write the whole method. If you think there is more than one possible correct answer, just answer one of them.

16. You are writing a method `findUpperCase` that takes as input a phrase and returns an array that contains all of the upper case letters in the input phrase. What does the *method header* look like? Just write the header. Do not write the whole method. If you think there is more than one possible correct answer, just answer one of them.

17. Which of the following are examples of *compile-time* errors? Write the *letters* corresponding to the correct answers on your answer sheet.

    (a) Accessing an array at index -1

    (b) Dividing an integer by 0

    (c) Passing an `int` value as input to a method that expects a `double` input argument

    (d) Accessing a variable outside of its scope.

    (e) Omitting a semi-colon at the end of a statement.

For the next two questions, consider the following block of code:

```
for(int i=0; i<5; i+=2){
  for(int j=0; j<=i; j++){
    System.out.println("Bubblegum");
  }
}
```

18. How many times is the condition in the *inner* loop evaluated?

19. How many times does the text `Bubblegum` get printed?

20. What are the elements referred to by the variable `a` after the following code executes?

```
public static void main(String[] args){
  int[] a = {1,2,3,4,5,6};
  mystery(a);
}

public static void mystery(int[] a){
  for(int i=0; i<a.length/2; i++){
    int temp = a[a.length-1-i];
    a[a.length-1-i] = a[i];
    a[i] = temp;
  }
}
```

21. What prints when the following code executes?

```
public static void main(String[] args){
  int[][] a = new int[4][];
  mystery(a);
  System.out.print(a.length + ", ");
  System.out.println(a[2].length);
}

public static void mystery(int[][] a){
  for(int i=0; i<a.length; i++){
    a[i] = new int[i+1];
  }
}
```

22. What prints when the following code executes?

```
public static void main(String[] args){
  String s = "word";
  mystery(s);
}

public static void mystery(String s){
  String t = "";
  for(int i=0; i<s.length(); i++){
    t = t + (char)((int)s.charAt(i) + i);
  }
  System.out.println(t);
}
```

23. What prints when the following method executes?

```
public static void method(){
  int[] a = {1,2,3,4,5};
  try{
    for(int i=0; i<=a.length; i++){
      System.out.print(a[i]/i + " ");
    }
    System.out.println();
  }catch(ArrayIndexOutOfBoundsException e){
    System.out.print("Bang! ");
  }catch(ArithmeticException e){
    System.out.print("Pow! ");
  }catch(Exception e){
    System.out.print("WHAM! ");
  }finally{
    System.out.print("Finally... ");
  }
}
```

Long Answer Question (32 points total)

For this question, you will use the booklets provided to write several methods. You are not permitted to use any String methods *except*:

- length()
- charAt()

You may not use any methods from the `Arrays` library.

In a class `AlphabetSoup` you will write the following methods:

- A method `countChar` that takes as input a `String` and a `char` and returns how many times that character is present in the String. Your method must be case sensitive (for example, if you are counting the letter 'a', DO NOT also include 'A' in your count).
- A method `haveChar` that takes as input a `String[]` and a `char` and determines whether or not every String in the array contains the character at least once. In order to get full marks you *must* use the `countChar` method defined above to do this.
- A method `copySubArray` that takes as input a `String[]`, a start index, and an end index, and returns a new array of appropriate length containing all elements in the old array from the start index (inclusive) to the end index (exclusive). You may assume

$$0 \leq start\_index < end\_index < array\ length$$

You will now use some of the methods above to write a program that does the following:

- It takes as input using command line arguments a single character, followed by an arbitrary number of Strings.
- If the first String in `args` is not a single character, or if the length of `args` is less than two, your program must throw an `IllegalArgumentException`.
- It then prints whether or not all of the input Strings contain the given character.

For example, if the input is `a cat dog aardvak` It should print `The letter a is not present in all of the Strings`. If the input is `a apple banana` It should print `The letter a is present in all of the Strings`.

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 int length()`: Calculates the length of this `String`.
  - `public char charAt(int i)`: Gets the `char` at position `i` of the `String`. Note that counting starts from 0 so that to get the first character of the `String` you should input `i` equals 0.
  - `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 String substring(int start, int finish)`: Returns a new String composed of the `this` String starting from index `start` and up to, but not including index of `finish`
  - `public String replace(char c, char d)` : Returns a new String with all occurrences of the character `c` in the `this` String replaced by the character `d`.
  - `public char[] toCharArray()`: Converts this `String` to a new character array.
- `PrintStream` (package `java.io`) Methods:
  - `public void print(boolean b)`: Prints `boolean` value `b`.
  - `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(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 exp(double a)`: Returns Euler's number e raised to the power of `double` value `a`.