Net ID:_____ **Name:**_____

CSCI-UA.0101-004 – Midterm Exam #1

March 1st, 2016

Instructor: Joseph Versoza

Ask the person to your left for their first name (leave blank if next to empty seat or wall):

_____

Ask the person to your right for their first name (leave blank if next to empty seat or wall):

_____

Keep this test booklet closed until the class is prompted to begin the exam

- Computers, calculators, phones, textbooks or notebooks are **not allowed** during the exam
- Please turn off your phone to avoid disrupting others during the exam

Instructions

- The code samples use an abbreviation, `syso`, for `System.out.println`
- You can use the same abbreviation when writing your own code.
- Depending on the question, you may or may not need to define a class and / or a main method. **Read the instructions carefully**.
- A reference can be found in the last page.
- The last page can also be used as scrap paper.

Choose 3 of the 1st 4 questions (#'s 1 through 4) to complete. Each question is worth 12 points.

Use the remaining question as extra credit (worth 1/2 of the points earned). **Specify which question is extra credit by writing EXTRA CREDIT** in caps above the question. If you do not specify extra credit, the last question (#4) will be counted as extra credit.

1. Answer the questions about the code on the left in the column on the right.

   (a) Assume that all of the code is within the **`main`** method of a Java program
   (b) Additionally, assume that a **`Scanner`** object called **`input`** is within scope
   (c) **Show your work** (where possible) **for partial credit** (for example, write the values of variables, add comments in the code inline, etc.)

Code	Question #1

`char ch = '\u0051';`
`int i = 0b00001110;`
`syso(ch);`
`syso(i);`
	What is the **output** of this program? **Show** your **work** for **partial credit**.
	
	`Q`
	`14`

`int total = 0;`
`for(int i = 10; i > 2; i -= 2) {`
`  syso(i);`
`  total += i;`
`}`
`syso(total);`
	What is the **output** of this program? **Show** your **work** for **partial credit**.
	`10`
	`8`
	`6`
	`4`
	`28`

`int num = input.nextInt();`
`String medal = "";`
`switch(num) {`
`  case 1:`
`  medal = "gold";`
`  case 2:`
`  medal = "silver";`
`  case 3:`
`  medal = "bronze";`
`}`
`syso(medal);`
	The program is meant to display the medal corresponding the number, num... for example, `1` should result in `gold`, and typing in `3` should display `bronze`. **There is a logical error** in the code. **Fix the error** by marking-up the code on the left (cross out, add lines, draw arrows).
	
	`Add break;`

`int x, y;`
 	
`do {`
 	
`  syso("Please enter two numbers");`
 	
`  x = input.nextInt();`
 	
`  y = input.nextInt();`
 	
`} while(x > y && x >= 0 && y >= 0);`
	Convert the `do while` loop into a ***regular*** **`while`** loop. You can mark-up the code on the left (cross out, add lines, draw arrows) to do this.
	`int x = 1, y = 1;`
	
	`while(x > y && x >= 0 && y >= 0) {`
	
	`  syso("Please enter two numbers");`
	
	`  x = input.nextInt();`
	
	`  y = input.nextInt();`
	
	`}`

2. And mistakes were made. The following two parts deal with syntax and logic errors.

Part 1 – Syntax / Compile Time Errors

Circle 4 unique syntax / compile errors in the following two method definitions (assume that both methods exist in a class) and give a short, 2 or 3 word description of the error. Draw arrows to help annotate.

```
public static void foo(int bar) {    // <---- 1. overloading

      int[] numbers;

      int x = 12  ;           // <---- 2. missing semicolon

      numbers = {x, 2, 3};  // <---- 3. array initialization w/ {} on 1 line

}

public static int foo(int bar) {// <---- 1. overloading

      if(bar > 1) {
            int baz = 0;
      } else {
            int baz = 25.0; //<---------- 4. double and int
      }

      return bar + baz;          // <--- 5. bar and baz not in scope
}
```

Part 2 – Logic Errors

Fix 2 logic errors in the sort method below. Use arrows, cross-outs and code to show your fixes:

```
public static void sort(int[] numbers) {

      for(int i = 0; i < numbers.length - 1; i++) {

            int min = 0; // <---- equal to i

            for(int j = i + 1; j < numbers.length; j++) {

                  if(numbers[j] < numbers[min]) {
                        min = j;
                  }

            }

            if(min != i) {
                  // \/\/\/\/ incorrect swap
                  numbers[min] = numbers[i];
                  numbers[i] = numbers[min];

            }
      }
}
```

3. **Part 1 – True or False**

(a) ___T___ `(!Character.isUpperCase('a'))`

(b) ___T___ `(false && true || 5 != 2)`

(c) __F____ `.java` files contain bytecode and `.class` files contain source code

(d) __T____ the class name in a Java program must match the name of the file that it's defined in

(e) __T____ a variable declared in a for loop is available (in scope) until the end of the for loop block

(f) __T____ the continue statement skips the remainder of the body of a loop and goes to the next iteration of the loop

Part 2 – What's the Output?

Write the output of the following code in the space provided.

`char ch = 'g';`
`syso(ch--);`
`syso(ch);`
 	
`g`
`f`
	`int x = 5;`
	`syso(++x);`
	`syso(x);`
	
	`6`
	`6`

Part 3 – What's the Output?

Draw the output of the code on the left in the grid on the right. If you need to write a space character, just leave a box blank. You do not have to use all of the squares. (3 points)

```
int size = 5;
for (int i = size; i > 0; i--) {

  for (int j = i ; j < size; j++) {
    System.out.printf("-");
  }

  for (int j = 1; j <= i; j++) {
    System.out.printf("%s", j);
  }

  System.out.println(); // print newline
}
```

1	2	3	4	5		
-	1	2	3	4		
-	-	1	2	3		
-	-	-	1	2		
-	-	-	-	1		

4. Short answer questions.

(a) What is the output of the following code? Error or no output are possible.

```
int i = 3;
double d = i;
syso(d);            3.0
```

(b) What is the output of the following code? Error or no output are possible.

```
double num1 = 7;
long num2 = num1;
syso(num2);         error
```

(c) What is type casting?

Converting from one type to another

(d) When is an explicit type cast required?

Narrowing conversions

(e) In the context of type casting, what is meant by widening a type and narrowing a type?

Narrowing – going from a type with a greater range of values to smaller range of values
Widening – going from a type with a smaller range of values to a greater range of values

(f) Whys does floating point arithmetic yield results that look slightly inaccurate?

Some floating point numbers can't be represented in a finite way in binary

(g) Describe one way to mitigate floating point arithmetic issues.

Use type with larger range of values, use BigDecimal class, add from smallest to largest

(h) Name the two broad categories of types... and give an example of a type in each category.

reference – int[][] foo
primitive – 5.0

(i) Why must you be careful when passing an Array as an argument to a method? What is *actually* being passed?

The value passed is a reference to the Array in memory (so changes made on the Array in the method are "seen" outside of the method)

Choose 4 of the last 5 questions (#'s 5 through 9) to complete. **Specify which question you are skipping by writing DO NOT GRADE** in caps above the question. If you do not, I will count the last question as the question you are skipping.

5. Write a **complete program**, **including** the **class definition**,the **`main`** method definition, and any necessary **imports**. The program should output the **average of all of the digits in a `String`** provided by the user.

   (a) ask the user for a `String`
   (b) add all of the numbers that are contained in the String; ignore everything else
   (c) treat each character as a single digit (for example, 12 is a 1 and 2, not a single 12)
   (d) output the average of all numbers entered with **2 decimal places**
   (e) **hint**: you can do this with or without unicode code points
   (f) **hint**: similar to the `BinHexDec.java` homework, you can use `Character.getNumericValue` to get the int value that a single character represents – example usage of the method is as follows: `Character.getNumericValue('5') // returns 5`
   (g) **hint**: alternatively, you can use Integer.parseInt, but you'll have to figure out how to extract a single string from the user input
   (h) sample user interaction below:

```
Please enter a string:
> 12hello3
The average of the digits in the string is 2.00
```

```
+1 imports
+1 class name
+1 main
+2 scanner and syso
+1 input
+2 loop over string (for loop, length of string)
+1 use charat
+2 check if it's a digit (if statement, correct condition)
+1 if it's a digit get numeric value
+2 keep track of total and count
+2 print out average w/ 2 decimal places
```

```
import java.util.Scanner;
public class JustYourAverageQuestion {
	public static void main(String[] args) {
		Scanner input = new Scanner(System.in);
		System.out.println("Please enter a string:");
		String s = input.next();
		char ch;
		int total = 0, count = 0;
		for(int i = 0; i < s.length(); i++) {
			ch = s.charAt(i);
			if(Character.isDigit(ch)) {
				total += Character.getNumericValue(ch);
				count += 1;
			}
		}
		// watch out for divide by 0! Also, divide by double...
		if(count > 0) {
			System.out.printf(
				"The avg of the digits in the string is %.2f",
				total / count / 1.0 );
		}
	}
}
```

6. Write a **method** that **moves every** instance of the character, `ch`, in an `Array` of `chars` to the **end** of the `Array`. You do not have to specify the class that this method is in, and you do not have to write a a `main` method (assume both already exist).

```
public static void moveToEnd(char[] characters, char ch)
```

(a) move every character that's the same as the second parameter, `ch`, to the end of the `Array`
(b) maintain the relative order of all of the other characters
(c) do this in place; that is... when you create your method, don't return anything, just change the incoming `Array`
(d) the original `Array` of `chars`, characters, should remain unchanged if `ch` is not in it
(e) for example, with exclamation point (!) as the second argument:

```
someCharacters = ['!', 'h', 'e', '!', '!', 'y'];

// call with exclamation point
moveToEnd(someCharacters, '!');

// prints out  ['h', 'e', 'y', '!', '!', '!']
syso(someCharacters);
```

```
+1 method header (given!)
+3 outer loop to repeat process: (1) loop, (3) either loop over all k
elements or count swaps / boolean
+3 inner loop to go through every position:(1) loop (2) skip last
+3 check if char matches: (1) if statement, (2) check if character
+3 swap: (1) temp var, (2) swap
+2 changes original
+1 no return!
```

```
public static void moveToEnd1(char[] arr, char ch) {
      boolean keepSorting = true;
      while(keepSorting) {
            keepSorting = false;
            for(int i = 0; i < arr.length - 1; i++) {
                  if(arr[i] == ch && arr[i + 1] != ch) {
                        arr[i] = arr[i + 1];
                        arr[i + 1] = ch;
                        // or
                        //char tmp = arr[i];
                        //arr[i] = arr[i + 1];
                        //arr[i + 1] = tmp;
                        keepSorting = true;
                  }
            }
      }
}

public static void moveToEnd2(char[] arr, char ch) {
      for(int i = arr.length - 1; i >= 0; i--) {
            if(arr[i] == ch) {
                  for(int j = i; j < arr.length - 1; j++) {
                        // System.out.printf("%s, %s, %s\n", i, j,
Arrays.toString(arr));
                        char temp = arr[j];
                        arr[j] = arr[j + 1];
                        arr[j + 1] = temp;
                  }
            }
            // System.out.println(Arrays.toString(arr));
      }
}
```

```
public static void moveToEnd3(char[] arr, char ch) {
      for(int i = 0; i < arr.length - 1; i++) {
            int min = i;
            for(int j = i + 1; j < arr.length; j++) {
                  if(arr[j] != ch && arr[i] == ch) {
                        min = j;
                        break;
                  }
            }
            if(min != i) {
                  char temp = arr[i];
                  arr[i] = arr[min];
                  arr[min] = temp;
            }
      }
}
```

7. Write a **method** called **`lastIndexOf`**. It will give back the index of the **last occurrence** of a `String` in an `Array` of `Strings` that comes **before** the index specified by `endIndex`. Do not use the string method, `lastIndexOf`. You do not have to specify the class that this method is in, and you do not have to write a a `main` method (assume both already exist).

   `public static int lastIndexOf(String[] haystack, String needle, int endIndex)`

   (a) `haystack` is the `String Array` to search, `needle` is the `String` to search for
   (b) give back the index of the last occurrence of `needle` in `haystack` (`String` in `Array`) that appears before `endIndex`
   (c) give back `-1` if `needle` is not in `haystack` (`String` is not found in `String Array`) or if needle appears at or after `endIndex`
   (d) assume `endIndex` is a positive integer; method should still work if `endIndex` > length of `Array`
   (e) example usage below:

```
String[] words = {"hello", "hi", "what's up?", "hi", "yo"};
int a = lastIndexOf(words, "hi", 4);      // a is  3
int b = lastIndexOf(words, "hi", 3);      // b is  1 (before index 3!)
int c = lastIndexOf(words, "hello, 10"); // c is  0
int d = lastIndexOf(words, "bye", 4);     // d is -1
```

+1 method header
+2 handle 0 for end index
+2 handle end index larger than length
+6 loop backwards, starting at end – 1, (2) -- (2) start at end (2) >= 0 or (-3 return 1st rather than last)
+1 set index
+1 default to -1
+2 break
+1 return

```
public static int lastIndexOf1(String[] haystack, String needle, int end) {
      int index = -1;
      if(end <= 0) {
            return -1;
      }
      if(end > haystack.length) {
            end = haystack.length;
      }
      for(int i = end - 1; i >= 0; i--) {
            if(haystack[i].equals(needle)) {
                  index = i;
                  break;
            }
      }
      return index;
}
```

```
public static int lastIndexOf2(String[] haystack, String needle, int end) {
	int index = -1;
	if(end <= 0) {
		return -1;
	}
	if(end > haystack.length) {
		end = haystack.length;
	}
	for(int i = 0; i < end; i++) {
		if(haystack[i].equals(needle)) {
			index = i;
		}
	}
	return index;
}
```

8. Write a **complete program**, **including** the **class definition**,the **`main`** method definition, and any necessary **imports**. The program should:

   (a) continually ask the user for a number
   (b) stop when the number is 0
   (c) output the largest number and the smallest number
   (d) do not count 0 as one of the numbers to consider for largest and smallest
   (e) negative numbers are allowed
   (f) do not use an array to do this
   (g) example interaction (characters after the greater than sign, >, is user input):

```
Give me a number > 7
Give me a number > -3
Give me a number > 12
Give me a number > 0

The largest number is 12
The smallest number is -3
```

```
import java.util.Scanner;
public class MaxMin {
	public static void main(String[] args) {
		Scanner input = new Scanner(System.in);
		System.out.println("Please enter a number\n> ");
		int num = input.nextInt();
		if(num != 0) {
			int max = num;
			int min = num;
			while(num != 0) {
				System.out.println("Please enter a number\n> ");
				num = input.nextInt();
				if(num == 0) {
					break;
				}
				if(num > max) {
					max = num;
				}
				if(num < min) {
					min = num;
				}
			}
			System.out.printf("The max is %s, the min is %s", max, min);
		}
	}
}
```

9. Write a **method** called **`shift`**, which will give back a new `Array` by **shifting** all of the **elements** in the original `Array` **to the right**. You do not have to specify the class that this method is in, and you do not have to write a a `main` method (assume both already exist).

   (a) it should have **two parameters**, an **`Array` of `int`s**, and an **`int`** representing an offset
   (b) it should **return** a new **`Array`**
   (c) assume that the `Array` passed in is not empty
   (d) assume that that the Array will have at least `2` elements, and the offset is at least `0`
   (e) shift every element to the right for that many positions
   (f) if an element is shifted *out* of the `Array`, move it to the beginning or the end of the `Array`
   (g) example output below – `Array` and offset are on the left, `Array` returned is on the right after `-->`

```
[1, 2, 3, 4, 5], 1 --> [5, 1, 2, 3, 4] // shift all elements 1 to the right
[1, 2, 3, 4, 5], 2 --> [4, 5, 1, 2, 3] // shift all elements 2 to the right
[1, 2, 3, 4, 5], 8 --> [3, 4, 5, 1, 2] // shift all elements 8 to the right
```

```
public static int[] shift(int[] arr, int offset) {
	int[] shifted = new int[arr.length];
	int newIndex;
	// handles shift greater than array length
	offset = offset % arr.length;
	for(int i = 0; i < arr.length; i++) {
		newIndex = i - offset;
		if(newIndex < 0) { // handle negative index
			newIndex = arr.length + newIndex;
		}
		shifted[i] = arr[newIndex];
	}
	return shifted;
}

public static int[] shift2(int[] arr, int offset) {
	int[] shifted = new int[arr.length];
	// handles shift greater than array length
	offset = offset % arr.length;
	for(int i = 0; i < shifted.length; i++) {
		if(i < offset) {
			shifted[i] = arr[shifted.length - offset];
		} else {
			shifted[i] = arr[i - offset];
		}
	}
	return shifted;
}

public static int[] shift3(int[] arr, int offset) {
	int[] shifted = new int[arr.length];
	for(int i = 0; i < arr.length; i++) {
		shifted[(i + offset) % arr.length] = arr[i];
	}
	return shifted;
}
```