Name (print): ________________________________

Instructions

• Keep your eyes on your own paper and do your best to prevent anyone else from seeing your work.
• Do NOT communicate with anyone other than the professor/proctor for ANY reason in ANY language in ANY manner.
• This exam is close notes, close books, and no calculator.
• Turn off all mobile devices and put them away now. You cannot have them on your desk.
• Write neatly and clearly indicate your answers. What I cannot read, I will assume to be incorrect.
• Remain in your seat until you are done. If you have a question raise your hand and wait until the instructor comes to you.

Time: This exam has 6 questions on 13 pages, including this title. Please make sure all pages are included. You will have 50 minutes to complete this exam; budget your time accordingly. Partial credit will be given, so if you are unable to complete a question at least give an attempt.

Honor Code: I commit to uphold the ideals of honor and integrity by refusing to betray the trust bestowed upon me as a member of the Emory community. I have also read and understand the requirements and policies outlined above.

Signature: ________________________________

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Q5</th>
<th>Q6</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1) (10 pts) For Question 1, consider the code below from the file Exam2q1.java

```
import java.util.Scanner;

public class Exam2q1 {
    public static int numInputs;

    public static double exp(double base, int pow) {
        double result = 1.0;
        int i = 0;
        while (i < pow) {
            result = result * base;
            i++;
        }
        return result;
    }

    public static double getBase() {
        System.out.print("Base? ");
        Scanner in = new Scanner(System.in);
        double b = in.nextDouble();
        numInputs++;
        return b;
    }

    public static int getPow() {
        System.out.print("Pow? ");
        Scanner in = new Scanner(System.in);
        int p = in.nextInt();
        numInputs++;
        return p;
    }

    public static void main(String[] args) {
        double base = getBase();
        int pow = getPow();
        double result = exp(base, pow);
        System.out.println("Result: "+result);
        System.out.println("numInputs: "+numInputs);
    }
}
```
a) List the names of all local variables in the program above (If there are multiple local variables with the same name, you can list them just once)

result, i, in, b, p, base, pow

b) List the names of all parameters variables in the program above (If there are multiple parameter variables with the same name, you can list them just once)

args, base, pow

c) List the names of the methods in the program above

exp, getBase, getPow, main

d) What would the output to the console be for the program if the user inputs 10 when prompted for the base and 2 when prompted for the pow?

Base? 10
Pow? 2
Result: 100
numInputs: 2
2) (14 pts) Searching and Sorting

Consider the following list for part a.

| 12 | 18 | 41 | 30 | 1 | 28 | 65 | 110 | 111 | 30 | 5 |

a) Write out the values in the list that will be inspected during a sequential (linear) search for the value 65. If a number is accessed more than once, list it more than once.

12, 18, 41, 30, 1, 28, 65

Consider the following list for part b and c

| 1 | 5 | 12 | 18 | 28 | 30 | 30 | 41 | 65 | 110 | 111 |

b) Write out the values in the list that will be inspected during a binary search for the value 7

30, 12, 1, 5
c) The 3 sets of list below show the sequence of a list of numbers when it is sorted using one of the sorting algorithms we studied. For each set of lists label it with the algorithm that was used to sort the initial list. The first list in each sequence is the initial array

<table>
<thead>
<tr>
<th></th>
<th>List Sequence</th>
<th>Sorting Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>i.</td>
<td>18 2 35 1 50</td>
<td>Selection Sort</td>
</tr>
<tr>
<td></td>
<td>1 2 35 18 50</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 2 18 35 50</td>
<td></td>
</tr>
<tr>
<td>ii.</td>
<td>18 2 35 1 50</td>
<td>Bubble Sort</td>
</tr>
<tr>
<td></td>
<td>2 18 35 1 50</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2 18 1 35 50</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2 1 18 35 50</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 2 18 35 50</td>
<td></td>
</tr>
<tr>
<td>iii.</td>
<td>18 2 35 1 50</td>
<td>Insertion Sort</td>
</tr>
<tr>
<td></td>
<td>2 18 35 1 50</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 2 18 35 50</td>
<td></td>
</tr>
</tbody>
</table>
3) (22 pts) For this question fill in the class named Exam2q3 and put the methods from part a and b in this class

**a)** Fill in the method `reverse` below. The method should create a new array with the elements in reverse order. The array should not modify the array referenced by the parameter value. The method returns the reversed array as its return value. For this question you may NOT use the methods in the java.util.Arrays class.

**b)** Fill in the main method for the class. The main method should extract the command line arguments into an array. Using the method you wrote in part (a) print this array in reversed order. For this question you may NOT use the methods in the java.util.Arrays class.

```java
public class Exam2q3 {
    public static double [] reverse(double [] list) {
        // TODO: Write a method that reverse the order of the
        // list array and returns the reversed array
        // NOTE: The values in the parameter list must remain
        // unchanged!

        double [] rev = new double[list.length];

        for (int i=0; i < list.length; i++) {
            rev[list.length-1-i] = list[i];
        }

        return rev;
    }
}
```
public static void main(String [] args) {
    double [] argsList;

    // TODO: Set the values in argsList
    // the command line arguments

    argsList = new double[args.length];
    for (int i=0; i < args.length; i++) {
        argsList[i] = Double.parseDouble(args[i]);
    }

    double [] reversed = reverse(argsList);

    // TODO: print out the elements in the reversed array. Separate each element by either a space
    // or a new line

    for (int i=0; i < reversed.length; i++) {
        System.out.println(reversed[i] + " ");
    }
}
4) (18 pts) Write Java statements for each of the directions below.

   a) Use a single statement to declare an array of double values named `array1` and initialize it to hold 5 values: 6.2, 7.3, 5.4, 2.1, 1.0

   ```java
   double [] array1 = {6.2, 7.3, 5.4, 2.1, 1.0};
   ```

   b) Write a method named `nextToLast` that takes an array of doubles as a parameter. The method should set the next to last value of the array to 3.14. If the array only has one element, then the method should set the last value of the array to 3.14.

   ```java
   public static void nextToLast (double [] x) {
       if (x.length == 1) {
           x[0] = 3.14;
       } else {
           x[x.length - 2] = 3.14;
       }
   }
   ```
Write a method named `evenAverage` that takes an array of integers as a parameter. The method should calculate the average of all of the elements in the array with an even index. This calculation should be returned by the method.

```java
public static double evenAverage (int [] x) {
    double sum = 0.0;
    int nElems = 0;
    for (int i=0; i < x.length; i+= 2) {
        sum += x[i];
        nElems++;
    }
    if (nElems > 0) {
        return sum / nElems;
    } else {
        return 0;
    }
}
```
5) (16 pts) Write the output that the following code will generate to the console. Assume the code compiles as written.

```java
public class Exam2q5 {
    public static int var1 = 10;
    public static boolean var2 = false;

    public static int method1(int var1) {
        if (var2 == false) {
            var1 -= 5;
        } else {
            var1 += 5;
            return var1;
        }
        System.out.println("var1 in method 1 = " + var1);
        return var1;
    }

    public static void method2() {
        var1 = 2;
        var1 = Exam2q5.var1 - 3;
        System.out.println("var1 in method2 = " + var1);
    }

    public static boolean method3(int var1, boolean b) {
        var1 = Exam2q5.var1 - 1;
        System.out.println("var1 in method3 = " + var1);
        if (b == true && var1 > 5) {
            System.out.println("Good");
            return true;
        } else {
            System.out.println("Bad");
            return false;
        }
    }

    public static void main(String[] args) {
        int a = method1(var1);
        int var1 = 7;
        System.out.println("a = " + a);
        System.out.println("var1 = " + var1);

        boolean b = method3(var1, true);
        System.out.println("b = " + b);
        System.out.println("var1 = " + var1);

        method2();
    }
}
```
Console Output

var1 in method 1 = 5  
a = 5  
var1 = 7  
var1 in method3 = 9  
Good  
b = true  
var1 = 7  
var1 in method2 = -1
6) Two dimensional arrays (20 pts)

Background: The array diagonal refers to elements along a line extending from the top-left corner of the array to the bottom-right corner. For example:

If you have a 3x3 2D array defined as:
```
int [][] a = {
    {1, 2, 3},
    {4, 5, 6},
    {7, 8, 9}
};
```

The diagonal will be the values 1, 5, 9

For this question you may assume that the 2D array parameter is "square." This means that the width is the same as the length of the 2D array.

a) Write a method named `sumDiagonal` that takes a 2D array of integers as a parameter. The method should return the sum of all elements on the diagonal of the 2D array.

```java
public static int sumDiagonal ( int [][] x) {
    int sum = 0;
    for (int i=0; i < x.length; i++) {
        sum += x[i][i];
    }
    return sum;
}
```
b) Write a method named `sumWithoutDiagonal` that takes a 2D array of integers as a parameter. The method should return the sum of all elements in the array excluding the elements on the diagonal. You may use the `sumDiagonal` method from part (a) to help if needed. (Assume the methods are class methods within the same class)

```java
class Matrix {
    public static int sumWithoutDiagonal (int[][] x) {
        int sum = 0;
        for (int i=0; i < x.length; i++) {
            for (int j=0; j < x[i].length; j++) {
                sum += x[i][j];
            }
        }
        sum -= sumDiagonal(x);
        return sum;
    }
}
```