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 closed notes, closed books, and no calculator.
- Turn all mobile devices off 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.
- Stop writing when told to do so at the end of the exam. I will take 5 points off your exam if I have to tell you multiple times.
- Academic misconduct will not be tolerated. Suspected academic misconduct will be immediately referred to the Emory Honor Council. Penalties for misconduct will be a zero on this exam, an F grade in the course, and/or other disciplinary action that may be applied by the Emory Honor Council.

Time: This exam has 6 questions on 9 pages including the title page. Please check to make sure all pages are included. You will have 50 minutes to complete this exam.

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>Question</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Points:</td>
<td>12</td>
<td>10</td>
<td>20</td>
<td>18</td>
<td>20</td>
<td>20</td>
<td>100</td>
</tr>
<tr>
<td>Score:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1
1. (12 points) For each entry below, state whether the code is correct as written or has an error (either syntax or logical). If there is no error, write the output. If there is, show where the error is (you can circle the incorrect code) and explain why it is an error.

- **Snippet 1**

```java
int [] a = {3, 4, 6, 16};
for (int i = 0; i < a.length; i++) {
    if (a[i] % 2 == 0) {
        System.out.println("Y");
    } else {
        System.out.println("N");
    }
}
```

Output or error:

- **Snippet 2**

```java
public static int someMethod(int num){
    return (2*num);
}
public static void reset(int num){
    num = -1;
}
public static void main(String [] args){
    int x = 10;
    x = someMethod(x);
    System.out.println(x);
    reset(x);
    System.out.println(x);
}
```

Output or error:

- **Snippet 3**

```java
int [] a;
int start = 2;
for (int i = 0; i < a.length; i++){
    a[i] = start;
    start *= 2;
}
System.out.println(a[0]);
System.out.println(a[1]);
```

Output or error:
• Snippet 4

```java
public static void res(int[] arr)
{
    for (int i = 0; i < arr.length; i++)
    {
        arr[i] = -1;
    }
}

public static void main(String[] args)
{
    int[] ar = {1, 2, 3, 4};
    int[] ar2 = {11, 12, 13};
    res(ar);
    System.out.println(ar[0]);
    System.out.println(ar2[1]);
    System.out.println(ar[2]);
    System.out.println(ar2[0]);
}
```

Output or error:

• Snippet 5

```java
public static void fun(int num)
{
    System.out.println("Num = " + num);
}

public static void main(String[] args)
{
    int a = fun(10);
    System.out.println(a);
}
```

Output or error:

• Snippet 6

```java
boolean[] c = new boolean[3];
c[1] = false;
for (int i = 0; i < c.length; i++)
{
    System.out.println(c[i]);
}
```

Output or error:
2. (10 points) Consider the array:
\{0, 2, 6, 11, 11, 11, 45, 40, 12, 4, 50, 40\}

- (2 points) Is it possible to perform a **linear search** on this array? If not, explain why. If it is possible, search for the value 11 using a linear search and list the elements in order that we will inspect.

- (8 points) Consider the array:
\{1, 4, 2, 7, 5, 10, 3, 1, 11\}

Apply the **selection sort** on this array and show all sorting steps. First step has been done for you. Do not omit duplicate rows.

Step 1: \{1, 4, 2, 7, 5, 10, 3, 1, 11\}
Step 2: \{1, 4, 2, 7, 5, 10, 3, 1, 11\}

3. (20 points) Draw the array arr after the code below executes. Assume that each code snippet is executed separately.

- **Snippet 1**

```java
boolean [] arr = new boolean[6];
int [] a = {1, 2, 7, 9};
int [] b = {1, 4, 7, 10};
for (int i = 0; i < arr.length; i++){
    if (b[i] - a[i] == 0 || b[i] - a[i] == 1) {
        arr[i] = true;
    }
}
```
• Snippet 2
  1 int [] arr = new int[3];
  2 double [] ar = {
    3 {1.1, 2.1, 3.1},
    4 {2.0, 2.2, 2.7},
    5 {1.4, 2.5, 2.8}};
  6 for (int i = 0; i < arr.length; i++){
    7 double sum = 0;
    8 for (int j = 0; j < ar[i].length; j++) {
      9 sum += ar[i][j];
    10 }
    11 arr[i] = (int) sum;
  12 }

• Snippet 3
  1 int [] arr = {1, 4, 6, 8, 10, 14, 20};
  2 for (int i = 0; i < arr.length/2; i++) {
    3 int temp = arr[i];
    4 arr[i] = arr[arr.length - 1 - i];
    5 arr[arr.length - 1 - i] = temp;
  6 }

• Snippet 4
  1 int [] arr = new int[5];
  2 int [] ar1 = {1, 4, 2, 10, 1};
  3 int [] ar2 = {2, 8, 1, 4, 2};
  4 for (int i = 0; i < arr.length; i++) {
    5 if (((ar1[i] % 2 == 0 && ar2[i] % 2 == 0) ||
    6 (ar1[i] % 2 == 1 && ar2[i] % 2 == 1)) {
      7 arr[i] = 1;
    8 }
4. (18 points) Consider the following program:

```java
public class Exam1{
    public static int y = 10;
    public static void fun(int par){
        System.out.println(y);  // Position 2
        System.out.println(q3c1.y);  // Position 3
        double y = 4.99;
        System.out.println(q3c1.y);  // Position 4
        System.out.println(y);  // Position 5
        par = -1;
    }
    public static boolean [][] retEmpty(){
        boolean [][] a = new boolean[5][5];
        return a;
    }
    public static void main(String [] args){
        int z = 15;
        String str = "CS170";
        System.out.println(z);  // Position 1
        System.out.println(z);  // Position 6
        System.out.println(y);  // Position 7
        {
            String s1 = "CS";
            boolean y = false;
            {
                String s2 = "170";
                System.out.println(str);  // Position 8
                System.out.println(s1);  // Position 9
                str = "CS170003";
            }
            System.out.println(s1);  // Position 10
            System.out.println(s2);  // Position 11
            System.out.println(y);  // Position 12
        }
        System.out.println(str);  // Position 13
    }
}
```

1. (13 points) Determine what would be printed in the following positions. If a variable cannot be printed at a given point, write SCOPING ERROR.

- Position 1:
- Position 2:
- Position 3:
- Position 4:
- Position 5:
- Position 6:
- Position 7:
• Position 8:
• Position 9:
• Position 10:
• Position 11:
• Position 12:
• Position 13:

2. (2 points) What line number is the class variable defined on? What is the scope of the class variables?

3. (2 points) How many parameters and of what type does the method \texttt{fun} take as an input?

4. (1 points) What type of variable does \texttt{retEmpty} return?

5. (20 points) Write the following methods:

   • (10 points) Write a method called \texttt{findNumber} that, given an array of integers \texttt{arr} and integer number \texttt{number} will return how many numbers \texttt{number} there is in the array \texttt{arr}. If there is no \texttt{number}, it returns -1. For example:
     \texttt{findNumber}\{(1, 7, 10, 20, 7), 7\} returns 2 (two 7s in this array),
     \texttt{findNumber}\{(1, 3, 7, 9, 5), 2\} returns -1 (no 2 in this array)
• (10 points) Write a method called `shiftArray` that, given an array of integers `arr` will modify this array such that it shifts all elements in the array by one to the left. First element will become the last element, second element will become the first one etc.
For example:
call `shiftArray([1, 4, 10, 20, 5])` will modify the array to `{4, 10, 20, 5, 1},
call `shiftArray([4, 4, 7, 8, 10])` will modify the array to `{4, 7, 8, 10, 4}`.
6. (20 points) Write a method called `RowMinimums` that, given a two dimensional array `arr` will return an array with minimums from each row. For example, if `arr` is:

\[
\begin{bmatrix}
2 & 5 & 1 & 7 \\
10 & 0 & 1 & 2 \\
5 & 6 & 7 & 1 \\
10 & 4 & 2 & 1
\end{bmatrix}
\]

the method should return an array `{1, 0, 1, 1}`.