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 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 8 questions on 16 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>7</th>
<th>8</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Points:</td>
<td>9</td>
<td>24</td>
<td>10</td>
<td>14</td>
<td>9</td>
<td>12</td>
<td>12</td>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>Score:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1. (9 points) Variable types and kinds. Consider the following code:

```java
public class Code1 {
    static int p = -2;
    public static void oddManipulation(int a, int b, int c) {
        int temp = a;
        b = temp;
        a = b;
        System.out.println("a: " + a);
        for (int i = 0; i < 1; i++) {
            System.out.print(a + 2 + " ");
        }
        p = a + q;
    }
    static int q = 1;
    public static void main(String[] args) {
        int x = 10, y = 11;
        int z = 9 - x;
        oddManipulation(x, y, z);
        System.out.println("x: " + x + " y: " + y + " z: " + z);
        System.out.println("p: " + p + " q: " + q);
    }
}
```

Solution

1. (1 pt) List all local variables in the `oddManipulation` method.
   - `int temp, int i`

2. (1 pt) List all parameter variables in the `oddManipulation` method.
   - `int a, int b, int c`

3. (1 pt) List all class variables in the program above.
   - `int p, int q`

4. (3 pts) What would be the output of the program if parameters were passed by **value**?
   - `a: 10` // Read the method body carefully -> it is NOT a swap
   - `12 x: 10 y: 11 z: -1`
   - `p: 11 q: 1`
5. (3 pts) What would be the output of the program if parameters were passed by reference?

a: 10
12 x: 10 y: 10 z: -1
p: 11 q: 1
2. (24 points) For each code fragment below, provide the output of the code. If the code contains errors, find **ALL ERRORS** in that code fragment. Assume that the method Arrays.toString( arrayName) will return a String containing the elements of the parameter array delimited/separated by commas. The java library method Arrays.toString(params) are overloaded so it can accept a parameter array of any type (char[], int[], double[], boolean[] and etc).

E.g.

```
int[] x = {12, 23, 54, 11};
System.out.println(Arrays.toString(x));
```

will output 12, 23, 54, 11 to the terminal.

a) Code:

```java
public class Code2A {
    public static void main(String[] args) {
        double[] c = {12.0, 1.1, 3.1, 2.1};
        System.out.println(Arrays.toString(c));

        double[] d = {3.0, 2.0, 1.0};
        for (int k = d.length - 1; k > 0; k--) {
            d[k] = c[k];
        }
        System.out.println(Arrays.toString(d));
    }
}
```

Output:

(No error - 1 pt )
12.0, 1.1, 3.1, 2.1 ( 2 pts)
3.0, 1.1, 3.1 (3 pts)

b) Code:

```java
int[] arr = {1, 2, 0, 1, 9};
for (int i = 1; i < arr.length; i++) {
    if (arr[i] <= arr[i + 1]) {
        System.out.println(i);
    }
}
System.out.println(Arrays.toString(arr));
```

Output:

(This has Error - 1pt)
2
3
ArrayIndexOutOfBoundsException (due to arr[4] <= arr[5], but there is no arr[5].
Fix the error: change i < arr.length to i < arr.length - 1

4
c) Code:

```java
public class Code2C {
    public static void main(String[] args) {
        int[] a = {1, 4, 7};
        int[] b = new int[3];
        int[] c = new int[1];
        c = a;
        a = b;
        switch (a[0]) {
            case 0: System.out.println(Arrays.toString(b));
            case 1: System.out.println(Arrays.toString(c));
            default: System.out.println(Arrays.toString(a));
        }
    }
}
```

Output:
(No error - 1pt)
(Understand the effect of the break statement in switch/follow through - 1pt)
(Understand the swap of variable references - 1pt)
(Correct output - 3 pts)
0, 0, 0
1, 4, 7
0, 0, 0

D) Code:

```java
public class Code2D {
    public static void main(String[] c) {
        int[] arr = {1, 2, 0, 1, 9};
        for (int i = 1; i < arr.length; i++) {
            arr[i] = arr[i] * arr[i + 1];
        }
        System.out.println(Arrays.toString(arr));
    }
}
```

Output:
(This has Error - 1 pt)
ArrayIndexOutOfBoundsException (due to arr[4] <= arr[5], but there is no arr[5].
(5 pts)
Fix the error: change i < arr.length to i < arr.length - 1
Then the output would be:
1, 0, 0, 9, 9
3. (10 points) Algorithms:
Assume, as in question 2, that the method Arrays.toString( arrayName) will return a String containing the elements of the parameter array delimited/separated by commas.
Consider the following 2 arrays:
\[
\text{int[]} \ a = \{1000001, 145, 130, 131, 129, 121, 21, -20, -40\};
\]
\[
\text{int[]} \ b = \{-10, 11, 21, 65, 71, 70, 112, 123\};
\]

a) (2 pts) For each array \(a\) and \(b\), can the linear (sequential) search algorithm be performed on the array? If yes, list the elements in the order we will inspect when searching for value 21 in that array. If not, explain why.

Yes, they are both arrays.
Array a: (1pt)
1000001, 145, 130, 131, 129, 121, 21
(When we find the element, we return immediately!!)

Array b: (1pt)
-10, 11, 21

It is inefficient to iterate through the entire array, but it is not an error, of course.

b) (2 pts) For each array \(a\) and \(b\), can the binary search algorithm be performed on the array? If yes, list the elements in order we will inspect when searching for value 22 in that array. If not, explain why.

No, neither array is sorted! (1 pt for each array).
Arrays must be sorted if we want to perform binary search on them.
c) (6 pts) Sorting. What is the output of the following program?

Notice that there are S.o.p statements in the selection sort method.

```java
public class Sort {
    public static void selectionSort(char[] a) {
        for (int i = 0; i < a.length; i++) {
            int minIdx = i;
            for (int k = i + 1; k < a.length; k++) {
                if (a[k] < a[minIdx]) {
                    minIdx = k;
                }
            }
            char tmp = a[i];
            a[i] = a[minIdx];
            a[minIdx] = tmp;
            // There’s an S.o.p statement here!
            System.out.println(Arrays.toString(a));
        }
    }

    public static void main(String[] a) {
        char[] b = {'a', 'f', 'b', 'd', 'u'};
        System.out.println(Arrays.toString(b));
        selectionSort(b);
        System.out.println(Arrays.toString(b));
    }
}
```

Output:

```
a, f, b, d, u  // First S.o.p statement in the main method
a, f, b, d, u  // i = 0
a, b, f, d, u  // i = 1
a, b, d, f, u  // i = 2
a, b, d, f, u  // i = 3
a, b, d, f, u  // Second S.o.p statement in the main method
```

S.o.p statements in the main method: 0.5 pt each statement x 2 = 1 pt
Each S.o.p statement in the selectionSort method: 0.5 pt x 5 = 2.5 pts
Correctly sorted array each round of i: 0.5 pt x 5 = 2.5 pts
4. (14 points) Method overloading and variable scopes
What is the output of following Java program?

```java
public class Scope {
    public static int z = 10;
    public static String a = "10";

    public static void main(String[] args) {
        System.out.println(z); // 10 - z is the class var
        {
            System.out.println(z); // 10 - z is the class var
        }
        {
            System.out.println(z); // 10 - z is the class var
            boolean z = true;
            System.out.println(z + " = " + Scope.z + "?"); true = 10 ?
                // z is the local boolean var,
                // Scope.z is the class var
        }
        System.out.println(z(a + z));
        // 10101010 - a + z is "1010" as a String,
        // applying the z method, returning 10101010 and outputing
        z = 20;
        {
            double z = 10;
            z = z( (int) z);
            // z is the local variable of type double. (int z) cast z to 10,
            // z(10) selects the method accepting an int parameter, so z(10) returns 100,
            // assigning back to z makes it 100.0
            System.out.println(z); // 100.0
        }
        z = z( z + Integer.parseInt(a) );
        // z is the class variable now, value = 20 originally.
        // z + Integer.parseInt(a) returns 30 (20 + 10).
        // So z(30) = 30 * 30 = 900
        System.out.println(z); // 900

        z = 10;
        for (int z = 10; z < 1000; z += Scope.z) {
            // the z inside the for loop and loop condition is the local var z.
            // Scope.z refers to the class variable, whose value is 10
            // (changed by z = 10 in the previous line).
            if (z % 200 == 10) {
                // only output number whose remainder after division by 200 is 10
                // so the if is true for z = 10, 210, 410, 610, 810
                // Note that 10 % 200 == 10 is true.
                System.out.println(z); // local variable z
            }
        }
    }
}
```

9
System.out.println(z); // prints class variable z = 10

public static String z(String z) {
    return z + z;
}

public static int z(int z) {
    return z * z;
}

Output:
10
10
10
true = 10?
10101010
100.0
900
10
210
410
610
810
10

(Each output is 1 pt, except for 100.0 is 2 points)
5. (9 points) 2-D arrays

a) The following code is a Java program that compiles and runs. Please provide the output of the program (i.e. what values are printed to the console/terminal). Output:

Output:
1.0, 2.0
1.0
0.0, 1.0

(Observe that there must be a new line between 10.0, 2.0 and 1.0 (empty 2nd line)
public class Arr2D {
    public static void main(String[] args) {
        double[][] arr = new double[3][2];

        for (int k = 0; k < arr.length; k++) {
            for (int i = 0; i < arr[0].length; i++) {
                arr[k][i] = k + i;
            }
        }

        // arr becomes
        // arr = { {0.0, 1.0}
        // {1.0, 2.0}
        // {2.0, 3.0} }
        System.out.println(Arrays.toString(arr[1]));
        // Outputs the subarray representing row with index 1 (2 pts)
        // so arr[i] = {1.0, 2.0}
        for (int a = 0; a < arr[0].length; a++) { // a runs from 0 to 1 (arr[0].length = 2)
            for (int b = 0; b < a; b++) { // b runs from 0 to a - 1
                System.out.print(arr[b][a] + " ");
            }
            System.out.println();
        }
        // Outer loop: a = 0
        // Inner loop b = 0 b < a is false so terminate inner loop
        // Output the new line (S.o.println)
        // Outer loop: a = 1
        // Inner loop b = 0 b < a is true so enter the loop, output arr[0][1] (which is 1.0)
        // Inner loop b = 1 b < a is false, so terminate the inner loop
        // Output the new line (S.o.println)
        // Loop part 5 pts
        System.out.println(Arrays.toString(arr[0]));
        // Outputs the subarray representing row with index 0 (2 pts)
    }
}
6. (12 points) Methods (Write both header and method body)

   a) (6 pts) Write a complete method named `getShortestStr` that accepts an array of Strings, and returns the String with the shortest length. If the array does not contain any String, return null.

   ```java
   public static String getShortestStr(String[] arr) {
     if (arr.length == 0) {
       return null;
     }
     int idxShortestStr = 0;
     for (int i = 0; i < arr.length; i++) {
       if (arr[idxShortestStr].length() > arr[i].length()) {
         idxShortestStr = i;
       }
     }
     return arr[idxShortestStr];
   }
   
   public static boolean checkFor2And3(double[] arr) {
     boolean hasSeen2 = false;
     boolean hasSeen3 = false;
     for (int i = 0; i < arr.length; i++) {
       if (arr[i] == 2.0) {
         hasSeen2 = true;
       } else if (arr[i] == 3.0) {
         hasSeen3 = true;
       }
     }
     return hasSeen2 && hasSeen3;
   }
   ```

   b) (6 pts) Write a complete method named `checkFor2And3` that accepts an array of floating type numbers (double), and returns true if the array contains both numbers 2.0 and 3.0; otherwise, return false.

   Example:

<table>
<thead>
<tr>
<th>Method Call</th>
<th>Return</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>checkFor2And3({1, 8, 0, 2, 5})</td>
<td>false</td>
<td>Only 2 appears.</td>
</tr>
<tr>
<td>checkFor2And3({1, 0, 2, 3})</td>
<td>true</td>
<td>Both 2 and 3 appear.</td>
</tr>
<tr>
<td>checkFor2And3({0, 4, 6, 5, 7, 6})</td>
<td>false</td>
<td>Neither number 2 nor 3 appears.</td>
</tr>
</tbody>
</table>

   ```java
   public static boolean checkFor2And3(double[] arr) {
     boolean hasSeen2 = false;
     boolean hasSeen3 = false;
     for (int i = 0; i < arr.length; i++) {
       if (arr[i] == 2.0) {
         hasSeen2 = true;
       } else if (arr[i] == 3.0) {
         hasSeen3 = true;
       }
     }
     return hasSeen2 && hasSeen3;
   }
   ```

   Solution:

   Each problem:
   - Header: 2 pts
   - Algorithm: 3 pts
   - Return value: 1 pt
7. (12 points) Methods (Write both header and method body)

a) (9 pts) Write a complete method named `numCommon` that accepts 2 arrays of integers, and returns the number of elements appearing in both arrays. Assume for each array, each element appears only once.

Hint: Use nested loops!

For example:

<table>
<thead>
<tr>
<th>Method Call</th>
<th>Return</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>numCommons({1, 8, 5}, {2, 9, 8})</td>
<td>1</td>
<td>Only 8 appears in both arrays.</td>
</tr>
<tr>
<td>numCommons({0, 9, 5, 2}, {1, 2, 9, 0})</td>
<td>3</td>
<td>0, 2 and 9 appear in both arrays</td>
</tr>
<tr>
<td>numCommons({1, 2, 11}, {9})</td>
<td>0</td>
<td>No element appears in both arrays.</td>
</tr>
</tbody>
</table>

```java
public static int numCommons(int[] a, int[] b) {
    int count = 0;
    for (int i = 0; i < a.length; i++) {
        for (int k = 0; k < b.length; k++) {
            if (a[i] == b[k]) {
                count++;
            }
        }
    }
    return count;
}
```

b) (3 pts) Write a complete method named `numNotCommon` that accepts 2 arrays of integers, and returns the number of elements appearing only one array both not the other; that is the total of elements appearing only in the first array but not the second, and elements appearing only in the second array but not the first.

You can (should) use a method call to `numCommon` from part a. You can assume that the method `numCommon` works as described.

For example:

<table>
<thead>
<tr>
<th>Method Call</th>
<th>Return</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>numCommons({1, 8, 5}, {2, 9, 8})</td>
<td>4</td>
<td>1, 2, 5, 9 are not common elements.</td>
</tr>
<tr>
<td>numCommons({0, 9, 5, 2}, {1, 2, 9, 0})</td>
<td>2</td>
<td>1 and 5 are not common elements</td>
</tr>
<tr>
<td>numCommons({1, 2, 11}, {9})</td>
<td>4</td>
<td>1, 2, 9, 11 are not common elements.</td>
</tr>
</tbody>
</table>

```java
public static int numNotCommon(int[] a, int[] b) {
    return a.length + b.length - 2 * numCommon(a, b);
}
```

**Solution:**

Part a:
- Header 2 pts
- Nested loops 4 pts
- Count variable 2 pt
- Return value 1 pt

Part b:
- Header: 1 pt
- Algorithm: 1 pt
- Return value: 1 pt
8. (10 points) 2D array

Write a complete method named `getTotals` that accepts a two-dimensional array of integers and also a boolean parameter `useRows`. If `useRows` is true, the method should return a new array containing the sums of elements in each row. If `useRows` is false, the method should return a new array containing the sums of elements in each column.

Example:

If a = { { 1, 2, 3, 4 },
        { 2, 5, 1, 7 } }

<table>
<thead>
<tr>
<th>Method Call on a</th>
<th>Returned array</th>
</tr>
</thead>
<tbody>
<tr>
<td>getTotals(a, true)</td>
<td>{10, 15}</td>
</tr>
<tr>
<td>getTotals(a, false)</td>
<td>{3, 7, 4, 11}</td>
</tr>
</tbody>
</table>

Solution 1:

```java
public static int[] getTotals(int[] arr, boolean useRows) {
    int[] resultArr;
    if (useRows) {
        resultArr = new int[arr.length];
    } else {
        resultArr = new int[arr[0].length];
    }
    for (int row = 0; row < arr.length; row++) {
        for (int col = 0; col < arr[0].length; col++) {
            if (useRows) {
                resultArr[row] += arr[row][col];
            } else {
                resultArr[col] += arr[row][col];
            }
        }
    }
    return resultArr;
}
```

Solution 2:

```java
public static int[] getTotals(int[] arr, boolean useRows) {
    if (useRows) {
        int[] resultArr = new int[arr.length];
        for (int row = 0; row < arr.length; row++) {
            for (int col = 0; col < arr[0].length; col++) {
                resultArr[row] += arr[row][col];
            }
        }
        return resultArr;
    } else {
        int[] resultArr = new int[arr[0].length];
        for (int row = 0; row < arr.length; row++) {
            for (int col = 0; col < arr[0].length; col++) {
                resultArr[col] += arr[row][col];
            }
        }
        return resultArr;
    }
}
```
Solution:
Method header: 3 pts
Algorithm - nested loops: 4 pts
Correct indices: 2 pts
Return value: 1 pt
**Bonus question** (+3 pts):
What would be the output of the following program?

```java
public class Bonus {
    int a = 10;
    a = a++ + a;
    int b = 10;
    b = b + b++;
    System.out.println(a + " and " + ++b);
}
```

Output:

---

**ASCII Table**

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUL</td>
<td>SOH</td>
<td>STX</td>
<td>ETX</td>
<td>EOT</td>
<td>ENQ</td>
<td>ACK</td>
<td>DEL</td>
</tr>
<tr>
<td>BS</td>
<td>HT</td>
<td>NL</td>
<td>VT</td>
<td>CR</td>
<td>SO</td>
<td>SI</td>
<td>DLE</td>
</tr>
<tr>
<td>DC1</td>
<td>DC2</td>
<td>DC3</td>
<td>DC4</td>
<td>NA</td>
<td>SYN</td>
<td>ETB</td>
<td>CAN</td>
</tr>
<tr>
<td>DG</td>
<td>SM</td>
<td>GS</td>
<td>FS</td>
<td>US</td>
<td>SP</td>
<td>SB</td>
<td>STX</td>
</tr>
<tr>
<td>0x10</td>
<td>0x11</td>
<td>0x12</td>
<td>0x13</td>
<td>0x14</td>
<td>0x15</td>
<td>0x16</td>
<td>0x17</td>
</tr>
<tr>
<td>0x00</td>
<td>0x01</td>
<td>0x02</td>
<td>0x03</td>
<td>0x04</td>
<td>0x05</td>
<td>0x06</td>
<td>0x07</td>
</tr>
<tr>
<td>0x08</td>
<td>0x09</td>
<td>0x0A</td>
<td>0x0B</td>
<td>0x0C</td>
<td>0x0D</td>
<td>0x0E</td>
<td>0x0F</td>
</tr>
<tr>
<td>0x10</td>
<td>0x11</td>
<td>0x12</td>
<td>0x13</td>
<td>0x14</td>
<td>0x15</td>
<td>0x16</td>
<td>0x17</td>
</tr>
<tr>
<td>0x18</td>
<td>0x19</td>
<td>0x1A</td>
<td>0x1B</td>
<td>0x1C</td>
<td>0x1D</td>
<td>0x1E</td>
<td>0x1F</td>
</tr>
<tr>
<td>0x20</td>
<td>0x21</td>
<td>0x22</td>
<td>0x23</td>
<td>0x24</td>
<td>0x25</td>
<td>0x26</td>
<td>0x27</td>
</tr>
<tr>
<td>0x28</td>
<td>0x29</td>
<td>0x2A</td>
<td>0x2B</td>
<td>0x2C</td>
<td>0x2D</td>
<td>0x2E</td>
<td>0x2F</td>
</tr>
<tr>
<td>0x30</td>
<td>0x31</td>
<td>0x32</td>
<td>0x33</td>
<td>0x34</td>
<td>0x35</td>
<td>0x36</td>
<td>0x37</td>
</tr>
<tr>
<td>0x38</td>
<td>0x39</td>
<td>0x3A</td>
<td>0x3B</td>
<td>0x3C</td>
<td>0x3D</td>
<td>0x3E</td>
<td>0x3F</td>
</tr>
<tr>
<td>0x40</td>
<td>0x41</td>
<td>0x42</td>
<td>0x43</td>
<td>0x44</td>
<td>0x45</td>
<td>0x46</td>
<td>0x47</td>
</tr>
<tr>
<td>0x48</td>
<td>0x49</td>
<td>0x4A</td>
<td>0x4B</td>
<td>0x4C</td>
<td>0x4D</td>
<td>0x4E</td>
<td>0x4F</td>
</tr>
<tr>
<td>0x50</td>
<td>0x51</td>
<td>0x52</td>
<td>0x53</td>
<td>0x54</td>
<td>0x55</td>
<td>0x56</td>
<td>0x57</td>
</tr>
<tr>
<td>0x58</td>
<td>0x59</td>
<td>0x5A</td>
<td>0x5B</td>
<td>0x5C</td>
<td>0x5D</td>
<td>0x5E</td>
<td>0x5F</td>
</tr>
</tbody>
</table>

Note that uppercase: $65 \leq x \leq 90$

Difference between A (65) and a (97) is 32!

Note that lowercase $97 \leq x \leq 122$

Difference between Q (81) and q (113) is 32!