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 7 questions on 12 pages including the title page. Please check to make sure all pages are included. You will have 75 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>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Points:</td>
<td>7</td>
<td>5</td>
<td>21</td>
<td>10</td>
<td>10</td>
<td>12</td>
<td>10</td>
<td>75</td>
</tr>
<tr>
<td>Score:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1. Parameter Passing. Consider the following code:

```java
public class Parameters {
    public static void swap(int a, int b) {
        int temp = a;
        a = b;
        b = temp;
    }
    public static void main(String[] args) {
        int x = 10;
        int y = 20;
        swap(x,y);
        System.out.println("x: "+x+" y: "+y);
    }
}
```

(a) (1 point) List all of the local variables in the program above.

**Solution:** x, y, temp

(b) (1 point) List all of the parameter variables in the program above.

**Solution:** a, b, args

(c) (1 point) List the names of all the methods in the program above.

**Solution:** swap, main

(d) (2 points) What would the code print if the parameters were passed by value?

**Solution:** a: 10 b: 20

(e) (2 points) What would the code print if the parameters were passed by reference?

**Solution:** a: 20 b: 10
2. Write Java statements for each of the directions below.

(a) (1 point) Create an array to hold 10 double values.

Solution: double[] x = new double[10];

(b) (1 point) Assign the value 6.9 to the last element of the array.

Solution: x[9] = 6.6;

(c) (1 point) Display the sum of the first two elements.

Solution: S.O.P.(x[0] + x[1]);

(d) (1 point) Create another array with the initial values 3.5, 5.5, 7.5 and 9.5.

Solution: double[] y = {3.5, 5.5, 7.5, 9.5};

(e) (1 point) Add the first element from the array created in part (a) to the first element of the array created in part (d) and assign the sum to a variable.

Solution: double z = x[0] + y[0];

3. For each of the parts below, give the output of the code.

(a) (4 points)

```java
public static String programming(String java, String python, String sql) {
    System.out.println("Programming in + sql " + java + " is harder than " + java);
    System.out.println("but " + python + " is the easiest.");
    return "java";
}

public static void main(String[] args) {
    String sql = "java";
    String python = "sql";
    String java = "python";

    String x = programming(python, java, sql);
    programming(x, java, "sql")
}
```

Solution: Programming in java is harder than sql but python is the easiest.
Programming in sql is harder than java but python is the easiest.
(b) (4 points)

```java
public static void s(int[] a, int i, int j) {
    int temp = a[i];
    a[i] = a[j];
    a[j] = temp;
}

int[] x = {1, 2, 3, 4};
int[] y = x;
s(x, 0, 3)
System.out.println(Arrays.toString(x))
s(y, 1, 2)
System.out.println(Arrays.toString(y))
```

Solution:

```
[4, 2, 3, 1]
[4, 3, 2, 1]
```

(c) (5 points)

```java
public class Shadows {
    public static int x;
    public static void method1(int x) {
        System.out.println("x 1: "+ x);
        x = Shadows.x;
        x = x + 1;
        System.out.println("x 2: "+ x);
    }

    public static void main(String[] args) {
        x = 7;
        System.out.println("x 3: "+ x);
        int x;
        x = 14;
        method1(x);
        System.out.println("x 4: "+ x);
        System.out.println("x 5: "+ Shadows.x);
    }
}
```

Solution:

```
x 3: 7
x 1: 14
x 2: 8
x 4: 14
x 5: 7
```
(d) (5 points)

```java
int[] list = {0, 2, 0, 3, 4};
for(int i = 0; i < list.length-1; i++) {
    if (list[i] != 0) {
        list[i] = list[i] + list[i+1];
    } else {
        int temp = list[i];
        list[i] = list[i+1];
        list[i+1] = temp;
    }
}
System.out.println(Arrays.toString(list));
```

**Solution:** [2, 0, 3, 4, 0]

(e) (3 points)

```java
int x = 3;
switch(x) {
    case 1:
        System.out.println("This is case 1");
        break;
    case 2:
        System.out.println("This is case 2");
    case 3:
        System.out.println("This is case 3");
    case 4:
        System.out.println("This is case 4");
        break;
    default:
        System.out.println("This is the default case");
}
```

**Solution:** This is case 3
This is case 4
4. Consider the array: 
\{2, 4, 7, 10, 11, 45, 50, 59, 60, 66, 69, 70, 79, 82, 91\}

(a) (2 points) List the elements in order that we will inspect when searching for the value 79 using a **binary search**.

**Solution:** 59, 70, 82, 79

would also accept 50, 70, 82, 79 which would be a length/2 implementation

(b) (2 points) List the elements in order that we will inspect when searching for the value 59 using a **sequential (or linear) search**.

**Solution:** 2, 4, 7, 10, 11, 45, 50, 59

Common error: not stopping once element was found, but continuing to examine more elements.

(c) (6 points) The array \{9, 4, 6, 2, 3\} can be sorted via different sorting algorithms including Selection Sort, Insertion Sort, and Bubble Sort. Label each sequence of sorting steps below with the name of the algorithm used to sort the initial array. The initial array has been repeated for you as **Step 1** for clarity.

i. Step 1: {9, 4, 6, 2, 3}
   Step 2: {4, 9, 6, 2, 3}
   Step 3: {4, 6, 9, 2, 3}
   Step 4: {2, 4, 6, 9, 3}
   Step 5: {2, 3, 4, 6, 9}

**Solution:** Insertion Sort

ii. Step 1: {9, 4, 6, 2, 3}
   Step 2: {4, 6, 2, 3, 9}
   Step 3: {4, 2, 3, 6, 9}
   Step 4: {2, 3, 4, 6, 9}
   Step 5: {2, 3, 4, 6, 9}

**Solution:** Bubble Sort

iii. Step 1: {9, 4, 6, 2, 3}
    Step 2: {2, 4, 6, 9, 3}
    Step 3: {2, 3, 6, 9, 4}
    Step 4: {2, 3, 4, 9, 6}
    Step 5: {2, 3, 4, 6, 9}

**Solution:** Selection Sort
5. (10 points) You are tasked with writing some code for the game show, The Price is Right. Write a method `priceIsRight` that takes an array of integers `bids` and an integer `price` as parameters. The method returns the element in the `bids` array that is closest in value to `price` without being larger than `price`. For example, if `bids` stores the elements 250, 450, 1000, then `priceIsRight(bids, 280)` should return 250, since 250 is the bid closest to 280 without going over 280. If all bids are larger than price, then your method should return -1.

You may assume there is at least 1 element in the array, and you may assume that the price and the values in bids will all be greater than or equal to 1.

The following table shows the results of some calls to your method:

<table>
<thead>
<tr>
<th>Method Call</th>
<th>returned value</th>
</tr>
</thead>
<tbody>
<tr>
<td>priceIsRight({900, 885, 989, 1}, 880)</td>
<td>1</td>
</tr>
<tr>
<td>priceIsRight({200}, 320)</td>
<td>200</td>
</tr>
<tr>
<td>priceIsRight({500, 300, 241, 99, 501}, 50)</td>
<td>-1</td>
</tr>
</tbody>
</table>

Solution:

```java
public static int priceIsRight(int[] bids, int price) {
    int max = -1;
    for(int i = 0; i < bids.length; i++) {
        if(max < bids[i] && bids[i] <= price) {
            max = bids[i];
        }
    }
    return max;
}
```

or something like:

```java
public static int priceIsRight(int[] bids, int price) {
    int correctBid = -1;
    int difference = price;
    for (int i = 0; i < bids.length; i++) {
        int d = price - bids[i];
        if (d < difference && bids[i] <= price) {
            difference = d;
            correctBid = bids[i];
        }
    }
    return correctBid;
}
```
}  

Scoring:
+1: function header (name, parameters, return type)
+2: variable initialization to correctly execute math
+2: uses loop correctly to examine all elements
+2: correctly deals with overbids (i.e., ignores them)
+1: calculates closest bid (max or difference) correctly
+2: returns correct value
6. (12 points) Write a function named `addElements` that takes in two arrays of doubles as parameters. Your function should add the corresponding elements of each array together and **return** the result in a new array. If the arrays are of unequal length, the method should simply use the elements in the longer array for the “unmatched” elements. Note that an array can be empty: `{}`.

Examples of function calls and returned arrays are below:

<table>
<thead>
<tr>
<th>Method Call</th>
<th>returned array</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>addElements({1, 5, 1}, {8, 3, 3})</code></td>
<td>{9, 8, 4}</td>
</tr>
<tr>
<td><code>addElements({1}, {5, 6, 7})</code></td>
<td>{6, 6, 7}</td>
</tr>
<tr>
<td><code>addElements({5, 3}, {})</code></td>
<td>{5, 3}</td>
</tr>
</tbody>
</table>

**Solution:**

```java
public static int[] addElements(int[] a, int[] b) {
    int max = 0;
    int min = 0;

    if (a.length > b.length) {
        max = a.length;
        min = b.length;
    } else {
        max = b.length;
        min = a.length;
    }

    int[] newArr = new int[max];
    int i;

    for (i = 0; i < min; i++) {
        newArr[i] = a[i] + b[i];
    }

    if (max != min) { //if arrays are different lengths, fill in from appropriate array
        if (a.length > b.length) {
            while (i < a.length) {
                newArr[i] = a[i];
                i++;
            }
        } else {
            while (i < b.length) {
```
newArr[i] = b[i];
    i++;
  }
}
return newArr;
}

Another solution which solves the problem differently:

public static int[] addElements(int[] a, int[] b) {
    int[] shorter;
    int[] longer;
    if (a.length > b.length) {
        shorter = b;
        longer = a;
    } else {
        shorter = a;
        longer = b;
    }

    int newArray = new int[longer.length];

    for (int i = 0; i < shorter.length; i++) {
        newArray[i] = shorter[i] + longer[i];
    }

    for (int i = shorter.length; i < longer.length; i++) {
        newArray[i] = longer[i];
    }

    return newArray;
}

Scoring:
+1 method header (name, parameters, return type)
+2 creates new array to store summed elements (ie doesn’t modify parameter arrays)
+3 uses iteration to correctly sums up elements from “matching” portions of arrays
+3 uses iteration to correctly copies elements from longer array if arrays are of unequal lengths
+1 correctly handles empty arrays
+2 returned array has correct values
(Since problem specified doubles but examples were integers, I gave credit for use of either datatype as long as usage was correct and consistent.)

Common errors: modifying an input array rather than making and returning a new array.
7. (10 points) The transpose of a matrix is one in which the rows have become columns and the columns have become rows. For example, the transpose of the matrix

\[
\begin{pmatrix}
a & b & c \\
d & e & f \\
g & h & i \\
\end{pmatrix}
\]

is the matrix

\[
\begin{pmatrix}
a & d & g \\
b & e & h \\
c & f & i \\
\end{pmatrix}
\]

Write a method `transpose` which takes a 2D array of integers as a parameter. The method should **not** modify the input parameter and should return a new, transposed 2D array of integers. Remember that matrices are not necessarily square (e.g. the number of rows will not necessarily be the same as the number of columns), but you may assume that the input matrix is not “ragged” (i.e., it will not have rows of differing lengths).

**Solution:**

```java
public static int[][] transpose(int[][] m) {
    int[][] x = new int[m[0].length][m.length];

    for(int row = 0; row < m.length; row++) {
        for (int col = 0; col < m[row].length; col++) {
            x[col][row] = m[row][col];
        }
    }

    return x;
}
```

**Scoring:**
+1 method header (name, parameters, return type)
+2 creates new 2D array
+1 new 2D array has correct dimensions
+2 iterates over each element
+2 correctly places element in place in new matrix
+2 returns new matrix

Common errors: not getting new matrix’s dimensions correct. Assuming matrix would only be square. Modifying input array rather than making a new one.