<table>
<thead>
<tr>
<th>Question</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Points</td>
<td>16</td>
<td>10</td>
<td>14</td>
<td>10</td>
<td>50</td>
</tr>
<tr>
<td>Score</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1. Consider the following:
   (a) (2 points) Write a statement that declares a one-dimensional int array named `arr` that has 25 elements.
   
   ```java
   int [] arr = new int[25];
   ```
   
   (b) (2 points) Write a statement that declares a two-dimensional int array named `arr` that has 5 rows and 20 columns.
   
   ```java
   int [][] arr = new int[5][20];
   ```
   
   (c) (2 points) True or False: In Java, the array index starts at 1.
   
   False
   
   (d) (3 points) Declare a method named `foo` that takes a one-dimensional integer array with the name `arr` as input and also has a return type of `void`. The method should set the last element of the array to -1. Assume the array has at least one element.
   
   Solution:
   ```java
   public static void foo(int [] arr) {
       arr[arr.length - 1] = -1;
   }
   ```
   
   (e) (3 points) Declare an integer array named `temp` of 1 element, invoke the `foo` method, and write the value of the last element of the array `temp`.
   
   ```java
   int [] temp = new int[1];
   foo(temp);
   ```
   
   -1
   
   (f) (4 points) What is the output of the code below:
   ```java
   int [] a = {2,6,8,10};
   int [] b = {2,4};
   int [] c = b;
   
   c = a;
   a[1] = 5;
   System.out.println(b[1]);
   System.out.println(c[1]);
   ```
   
   4
   5
2. (10 points) Consider the following code for binary search:

```java
public static int binarySearch(int[] list, int key) {
    int low = 0;
    int high = list.length - 1;

    while (high >= low) {
        int mid = (low + high) / 2;
        if (key < list[mid])
            high = mid - 1;
        else if (key == list[mid])
            return mid;
        else
            low = mid + 1;

        System.out.println(list[mid]);
    }
    return -1 - low;
}
```

Suppose we use binary search to look for the key 7 in the array `int [] list = {1,3,6,8,9,10,12}`. List the output for each iteration of the while loop on line 14. Show your work for partial credit.

```
8,3,6
```
3. Consider the function Mystery which is given below. This function has limitations on its input: all values in the array must be greater than 0 and there must be at least 1 element of each row.

```java
public static int[] Mystery(int[][] arr) {
    int[] nums = new int[arr.length];
    for (int i = 0; i < arr.length; i++) {
        int val = -1;
        for (int j = 0; j < arr[i].length; j++) {
            if (arr[i][j] > val) {
                val = arr[i][j];
            }
        }
        nums[i] = val;
    }
    return nums;
}
```

Answers the following questions:

(a) (2 points) What is the return type of the Mystery method?
   (a) int[]

(b) (2 points) What is the input parameter type and name for the Mystery method?
   (b) type = int[][], name = arr

(c) (3 points) Suppose we have the following variable:
   int[][] foo = {
       {1,1,3,9},
       {23, 21, 19},
       {9,45,8,4}
   };

   What is the return value for the call Mystery(foo)?
   (c) {9,23,45}

(d) (2 points) What is the data type of foo[0]?
   (d) int[]

(e) (2 points) What is the value of foo[1].length?
   (e) 3

(f) (3 points) Describe in words what the Mystery method does.

Solution: The method finds the max value for the $i^{th}$ row in the two-dimensional input array and stores it in the $i^{th}$ element of the nums (return) array.
4. (10 points) Write a method named `areBothEqual` that takes two 1-dimensional integer arrays as input and returns a boolean value of `true` or `false` if each element in both arrays are the same value. For example if the first array is `{1,2,3}` and the second array is `{1,2,3}` the method should return `true`; however, if the first array is `{3,2,1}` and the second array is `{1,2,3}`, the method should return `false`. Assume both input arrays have at least one element and are both the same length.

Solution:

```java
public static boolean areBothEqual(int[] arr1, int[] arr2) {
    for (int i = 0; i < arr1.length; i++) {
        if (arr1[i] != arr2[i])
            return false;
    }
    return true;
}
```