1 Short Answer (10 Points Each)

1. Write a linear search method for an integer array that takes in an array and target value as parameters and returns the first position of the target in the array. If the target is not in the array then the method should return -1.

   Solution:

```
public static int LinearSearch(int[] A, int N) {
    for (int index = 0; index < A.length; index++) {
        if (A[index] == N)
            return index;
    }
    return -1;
}
```

2. Write a method that does either the bubble sort, insertion sort or selection sort for an array of integers. You must also state which sort you are writing.

   Solution:

```
public static void BubbleSort(int A[]) {
    for (int i = A.length - 1; i > 0; i--) {
        for (int j = 0; j < i; j++) {
            if (A[j] > A[j + 1]) {
                int temp = A[j];
                A[j] = A[j + 1];
                A[j + 1] = temp;
            }
        }
    }
}

public static void InsertionSort(int[] A) {
    for (int itemsSorted = 1; itemsSorted < A.length; itemsSorted++) {
        int temp = A[itemsSorted];
        int loc = itemsSorted - 1;
        while (loc >= 0 && A[loc] > temp) {
            A[loc + 1] = A[loc];
            loc = loc - 1;
        }
        A[loc + 1] = temp;
    }
}

public static void SelectionSort(int[] A) {
    for (int lastPlace = A.length - 1; lastPlace > 0; lastPlace--) {
        int maxLoc = 0;
        for (int j = 1; j <= lastPlace; j++)
            if (A[j] > A[maxLoc])
                maxLoc = j;

        int temp = A[maxLoc];
        A[maxLoc] = A[lastPlace];
        A[lastPlace] = temp;
    }
}
```

3. Write a method that takes in a one-dimensional integer array as its only parameter and returns a one-dimensional integer array that is in the reverse order of the one that came in as a parameter.

   Solution:

```
public static int[] reverse(int[] A) {
    int[] B = new int[A.length];
    for (int i = 0; i < A.length; i++) {
        B[i] = A[A.length - 1 - i];
    }
    return B;
}
```

4. Write a method that takes as its only parameter a one-dimensional array (A) of doubles and returns a one-dimensional array (M) of doubles that has only two entries. The first entry of M is the minimum of the entries in A and the second entry of M is the maximum of the entries in A. If the length of the array A is 0 then the array M should have both of its entries as 0.

   Solution:

```
1  public static double[] MinMax(double[] A) {
2      double[] M = new double[2];
3      M[0] = 0;
4      M[1] = 0;
5      if (A.length > 0) {
6          double Max = A[0];
7          double Min = A[0];
8          for (int i = 0; i < A.length; i++) {
9              if (Min > A[i])
10                 Min = A[i];
11             if (Max < A[i])
12                 Max = A[i];
13         }
14         M[0] = Min;
15         M[1] = Max;
16     }
17     return M;
18 }
```

5. Write a method that takes a two-dimensional array of integers as its only parameter and returns a two-dimensional array that is the *transpose* of the original array. That is, if the input two-dimensional array has n rows and m columns then the output two-dimensional array will have m rows and n columns. The first row of the input array will be the first column of the output array, the second row of the input array will be the second column of the output array, and so on.

   So if the input array was,

```
 4   -5   -5   -1   -2
-1   -1   -3   -4   -3
 0   -3   -3    3    3
```

   The output array will be,

```
 4   -1    0
-5   -1   -3
-5   -3   -3
-1   -4    3
-2   -3    3
```

   Solution:

```
1  public static int[][] Transpose(int[][] A) {
2      int[][] B = new int[A[0].length][A.length];
3      int dim1 = A.length; // Gets the number of rows
4      int dim2 = A[0].length; // Gets the number of columns
5
6      for (int i = 0; i < dim1; i++) {
7          for (int j = 0; j < dim2; j++) {
8              B[j][i] = A[i][j];
9          }
10     }
11     return B;
12 }
```

6. Write a method that takes in two-dimensional array of integers and a target integer as its only two parameters. Search the two-dimensional array for the target. If it is found then return a one-dimensional array of two integers where the first entry is the row of the target value and the second is the column of the target. If the target is not found in the two-dimensional array the method should return -1 in both entries of the returned array.

   Solution:

```
1  public static int[] Search2D(int[][] A, int target) {
2      int[] Pos = new int[2];
3      Pos[0] = -1;
4      Pos[1] = -1;
5      for (int i = 0; i < A.length; i++) {
6          for (int j = 0; j < A[0].length; j++) {
7              if (A[i][j] == target) {
8                  Pos[0] = i;
9                  Pos[1] = j;
10             }
11         }
12     }
13     return Pos;
14 }
```

2 Program Traces (15 Points Each)

1. Write the output of the program in the Program Output box.

```
1  public class Thing {
2
3      private int a;
4      private double b;
5      public int c;
6
7      public Thing() {
8          a = 1;
9          b = 2;
10         c = 3;
11     }
12
13     public Thing(int x, int y, double z) {
14         a = x;
15         b = z;
16         c = y;
17     }
18
19     public void set(int x, double y, int z) {
20         a = x;
21         b = y;
22         c = z;
23     }
24
25     public String toString() {
26         return a + "  " + b + "  " + c;
27     }
28
29     public int DoSomething() {
30         a += c;
31         if (a % 3 == 1)
32             b = b / 2;
33         else
34             b = b + 2;
35         c *= 2;
36         return a + c;
37     }
38
39     public void DoSomethingElse(int d) {
40         if (d % 2 == 0){
41             a -= c;
42             b = b + a;
43             c--;
44         }
45     }
46 }
```

```
1  public class Trace1 {
2
3      public static void main(String[] args) {
4          Thing thing1 = new Thing();
5          Thing thing2 = new Thing(3, 5, 9);
6
7          System.out.println(thing1.toString());
8          System.out.println(thing2.toString());
9
10         thing2.DoSomething();
11         System.out.println(thing2.toString());
12
13         thing1.DoSomething();
14         System.out.println(thing1.toString());
15
16         thing1.set(4, 7, 11);
17         thing2.set(-1, 3, -7);
18
19         thing1.c = thing2.DoSomething();
20         System.out.println(thing1.toString());
21
22         thing2.DoSomethingElse(thing2.DoSomething()
               );
23         System.out.println(thing2.toString());
24     }
25 }
```

Solution:

```
1   2.0   3
3   9.0   5
8   11.0   10
4   1.0   6
4   7.0   -22
6   13.0   -29
```

2. Write the output of the program in the Program Output box.

```
1  public class Trace2 {
2
3      public static void println(int[] B) {
4          for (int i = 0; i < B.length; i++) {
5              System.out.print(B[i] + "  ");
6          }
7          System.out.println();
8      }
9
10     public static void DoSomething(int[] B) {
11         for (int i = 0; i < B.length; i++) {
12             B[i] = B[(i + 2) % B.length];
13         }
14     }
15
16     public static void DoSomething(int[] B, int k)
           {
17         for (int i = 0; i < B.length; i++) {
18             B[i] = B[i] * (i / 2) - k;
19         }
20     }
21
22     public static void DoSomething(int[] B, int[] C
           ) {
23         for (int i = 0; i < B.length; i++) {
24             if (i < C.length)
25                 B[i] = B[i] + C[i];
26             else
27                 B[i] = B[i] + C.length;
28         }
29     }
30
31     public static void set(int[] B) {
32         for (int i = 0; i < B.length; i++) {
33             B[i] = 3 * i + 1;
34         }
35     }
36
37     public static void set(int[] B, int k) {
38         for (int i = 0; i < B.length; i++) {
39             B[i] = (5 * i + 1) % k;
40         }
41     }
42
43     public static void main(String[] args) {
44         int[] A = new int[7];
45         int[] B = new int[5];
46
47         set(A);
48         println(A);
49         DoSomething(A);
50         println(A);
51         set(A);
52         DoSomething(A, 4);
53         println(A);
54         set(A);
55         set(B, 7);
56         DoSomething(A, A);
57         println(A);
58         println(B);
59     }
60 }
```

Solution:

```
1   4   7   10   13   16   19
7   10   13   16   19   7   10
-4   -4   3   6   22   28   53
2   8   14   20   26   32   38
1   6   4   2   0
```

3 Coding (20 Points)

In this exercise you will be completing the following program by coding the four methods, `print`, `populate`, `ColumnMaxAverage`, and `RowAverageMax`. Write your code in the code boxes on the next couple pages.

print This method is to take in as parameters a two-dimensional array of integers and an integer for the width of the print. The method will print the two-dimensional array to the screen using the input width for the columns.

populate This method is to take in as parameters a two-dimensional array of integers and two integers that designate the lower and upper bounds of the entries. The method will populate the entire array with values that are between the lower and upper bounds, inclusively.

ColumnMaxAverage This method will take in a single parameter of a two-dimensional array of integers calculate the maximum of each column and then average these maximums. The method will then return the average.

RowAverageMax This method will take in a single parameter of a two-dimensional array of integers calculate the average of each row and then return the maximum of those averages.

```
1  import java.util.Scanner;
2
3  public class Exam2Prog {
4
5      public static void print(int[][] A, int w) {
6          String format = "%" + w + "d";
7          for (int i = 0; i < A.length; i++) {
8              for (int j = 0; j < A[0].length; j++) {
9                  System.out.printf(format, A[i][j]);
10             }
11             System.out.println();
12         }
13     }
14
15     public static void populate(int[][] A, int l, int h) {
16         for (int i = 0; i < A.length; i++) {
17             for (int j = 0; j < A[0].length; j++) {
18                 A[i][j] = (int) (Math.random() * (h - l + 1)) + l;
19             }
20         }
21     }
22
23     public static double ColumnMaxAverage(int[][] A) {
24         double colmaxsum = 0;
25         for (int j = 0; j < A[0].length; j++) {
26             int colmax = A[0][j];
27             for (int i = 0; i < A.length; i++) {
28                 if (colmax < A[i][j])
29                     colmax = A[i][j];
30             }
31             colmaxsum += colmax;
32         }
33         return colmaxsum / A[0].length;
34     }
35
36     public static double RowAverageMax(int[][] A) {
37         double[] rowaverages = new double[A.length];
38         for (int i = 0; i < A.length; i++) {
39             int rowsum = 0;
40             for (int j = 0; j < A[0].length; j++) {
41                 rowsum += A[i][j];
42             }
43             rowaverages[i] = (double) rowsum / A[0].length;
44         }
45         double rowavgmax = rowaverages[0];
46         for (int i = 0; i < A.length; i++) {
47             if (rowavgmax < rowaverages[i])
48                 rowavgmax = rowaverages[i];
49         }
50         return rowavgmax;
51     }
```

```
52
53      public static void main(String[] args) {
54          Scanner kb = new Scanner(System.in);
55          System.out.print("Rows: ");
56          int r = kb.nextInt();
57          System.out.print("Cols: ");
58          int c = kb.nextInt();
59          System.out.print("Lower Bound: ");
60          int l = kb.nextInt();
61          System.out.print("Upper Bound: ");
62          int u = kb.nextInt();
63
64          System.out.println();
65          int[][] A = new int[r][c];
66          populate(A, l, u);
67          print(A, 4);
68          System.out.println();
69          System.out.println("Column Max Average = " + ColumnMaxAverage(A));
70          System.out.println("Row Average Max = " + RowAverageMax(A));
71      }
72  }
```