# 1 Short Answer (7 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 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 a two-dimensional array of doubles as its only parameter and returns a one-dimensional array of doubles that stores the column sums of the input array. That is, if the two-dimensional array has $n$ rows and $m$ columns then the one-dimensional array will have $m$ entries. The first entry will be the sum of the entries in column one, the second will be the sum of column two, and so on.

   **Solution:**

```
public static double[] ColSums(double[][] A) {
    double[] SumArray = new double[A[0].length];
    int dim1 = A.length; // Gets the number of rows
    int dim2 = A[0].length; // Gets the number of columns

    for (int i = 0; i < dim2; i++) {
        double sum = 0;
        for (int j = 0; j < dim1; j++) {
            sum += A[j][i];
```

```
10          }
11          SumArray[i] = sum;
12      }
13      return SumArray;
14  }
```

4. 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 }
```

5. Write a method that takes an arraylist of integers as its only parameter and returns another arraylist which is the "partial sums" of the input arraylist. Specifically, the first entry of the output list is the same as the first entry of the input list. The second entry of the output list is the sum of the first two entries of the input list. The third entry of the output list is the sum of the first three entries of the input list, and so on. For example, if the input arraylist looks like,

```
[10, 5, 2, 7, 12, 23, 45, 17]
```

   then the output arraylist would be,

```
[10, 15, 17, 24, 36, 59, 104, 121]
```

   **Solution:**

```
1  public static ArrayList<Integer> sums(ArrayList<Integer> A) {
2      ArrayList<Integer> B = new ArrayList<Integer>();
3
4      B.add(A.get(0));
5      for (int i = 1; i < A.size(); i++) {
6          B.add(A.get(i) + B.get(i - 1));
7      }
8
9      return B;
10 }
```

6. Write a method that will ask for an integer from the user and return the input integer. If the user inputs anything other then an integer then the program should display "Input was not an integer, please try again." and then ask for the input again. The method should not return until an integer has been input.

   **Solution:**

```
public static int getInteger() {
    Scanner kb = new Scanner(System.in);

    boolean inputNeeded = true;
    int value = 0;
    while (inputNeeded) {
        System.out.print("Input an integer: ");
        if (kb.hasNextInt()) {
            value = kb.nextInt();
            inputNeeded = false;
        } else {
            System.out.println("Input is not an integer, please try again.");
        }
        String clearbuf = kb.nextLine();
    }
    return value;
}
```

   or

```
public static int getInteger() {
    Scanner kb = new Scanner(System.in);

    boolean inputNeeded = true;
    int value = 0;
    while (inputNeeded) {
        System.out.print("Input an integer: ");
        try{
            value = kb.nextInt();
            inputNeeded = false;
        } catch(Exception e){
            System.out.println("Input is not an integer, please try again.");
        }
        String clearbuf = kb.nextLine();
    }
    return value;
}
```

7. Write a method called `reformat` that will takes an arraylist of integers and a single integer as parameters. The single integer will represent the number of columns for the output two-dimensional array. The method will output a two-dimensional array with the specified number of columns and as many rows as needed to store the arraylist in the two-dimensional array. The entries of the input arraylist will start across row one then go to row two and so on. If the array list does not fill the two dimensional array then the unused entries will be 0. For example, if our arraylist is the following,

   ```
   [10, 5, 2, 7, 12, 23, 45, 17]
   ```

   then using 2 columns gives,

   ```
   10    5
    2    7
   12   23
   45   17
   ```

   using 3 columns gives,

   ```
   10    5    2
    7   12   23
   45   17    0
   ```

   using 4 columns gives,

```
10    5    2    7
12   23   45   17
```

and using 5 columns gives,

```
10    5    2    7   12
23   45   17    0    0
```

**Solution:**

```
1  public static int[][] reformat(ArrayList<Integer> A, int cols) {
2      int rows = A.size()/cols;
3      int extra = A.size() % cols;
4      if (extra > 0)
5          rows++;
6      int[][] B = new int[rows][cols];
7
8      int row = 0;
9      int col = 0;
10     for (int i = 0; i < A.size(); i++) {
11         B[row][col] = A.get(i);
12         col++;
13         if (col == cols){
14             col = 0;
15             row++;
16         }
17     }
18
19     return B;
20 }
```

## 2 Program Traces (15 Points Each)

1. Write the output of the program.

```
1  public class Exam3Trace1 {
2
3      public static int[] DoSomething(int A[][]) {
4          int dim1 = A.length;
5          int dim2 = A[0].length;
6          int[] C = new int[dim1 * dim2];
7
8          int count = 0;
9          for (int i = 0; i < dim2; i++) {
10             for (int j = 0; j < dim1; j++) {
11                 C[count++] = A[j][i];
12             }
13         }
14
15         count = C.length - 1;
16         for (int i = 0; i < dim1; i++) {
17             for (int j = 0; j < dim2; j++) {
18                 A[i][j] = C[count--];
19             }
20         }
21
22         return C;
23     }
24
25     public static void PrintArray(int A[]) {
26         for (int i = 0; i < A.length; i++) {
27             System.out.print(A[i] + "  ");
28         }
29         System.out.println();
30     }
31
32     public static void PrintArray(int A[][]) {
33         int dim1 = A.length;
34         int dim2 = A[0].length;
35
36         for (int i = 0; i < dim1; i++) {
37             for (int j = 0; j < dim2; j++) {
38                 System.out.printf("%5d", A[i][j]);
39             }
40             System.out.println();
41         }
42     }
43
44     public static void main(String[] args) {
45         int[][] A = new int[3][5];
46         int[] B = new int[10];
47
48         for (int i = 0; i < 3; i++)
49             for (int j = 0; j < 5; j++) {
50                 A[i][j] = (i + 1) * (j + 1);
51             }
52
53         PrintArray(A);
54         System.out.println();
55         B = DoSomething(A);
56         PrintArray(A);
57         System.out.println();
58         PrintArray(B);
59     }
60 }
```

---

**Solution:**

```
    1    2    3    4    5
    2    4    6    8   10
    3    6    9   12   15

   15   10    5   12    8
    4    9    6    3    6
    4    2    3    2    1

1  2  3  2  4  6  3  6  9  4  8  12  5  10  15
```

2. Write the output of the program.

```
1  public class Thing {
2
3      private int[] stuff;
4      private int len;
5      private int max = 100;
6
7      public Thing(int l, int m) {
8          len = l;
9          max = m;
10         stuff = new int[len];
11     }
12
13     public int length() {
14         return len;
15     }
16
17     public void set(int i, int v) {
18         if (i < 0 || i >= len)
19             return;
20
21         if (v > max || v < -max) {
22             if (v > 0)
23                 v = max;
24             else
25                 v = -max;
26         }
27
28         stuff[i] = v;
29     }
30
31     public int get(int i) {
32         if (i < 0 || i >= len)
33             return 0;
34         else
35             return stuff[i];
36     }
37
38     public void mix(){
39         int temp = 0;
40         for (int i = 0; i < len; i++) {
41             temp = stuff[i];
42             stuff[i] = stuff[(3*i+1) % len];
43             stuff[(3*i+1) % len] = temp;
44
45             System.out.println(toString());
46         }
47     }
48
49     public String toString() {
50         String str = "";
51         for (int i = 0; i < len; i++) {
52             str = str + " " + stuff[i];
53         }
54         return str;
55     }
56
57 }
```

```
1  public class Exam3Trace2 {
2
3      public static void main(String[] args) {
4          Thing t = new Thing(7, 10);
5          t.set(0, 5);
6          t.set(3, -15);
7          t.set(1, 7);
8          t.set(6, -2);
9          t.set(10, 9);
10         t.set(4, 23);
11         System.out.println(t);
12         System.out.println();
13         System.out.println(t.length());
14         System.out.println(t.get(2));
15         System.out.println(t.get(4));
16         System.out.println(t.get(12));
17         System.out.println();
18         t.mix();
19         System.out.println(t);
20     }
21 }
```

---

**Solution:**

```
 5 7 0 -10 10 0 -2

7
0
10
0

 7 5 0 -10 10 0 -2
 7 10 0 -10 5 0 -2
 0 10 7 -10 5 0 -2
 0 10 7 -10 5 0 -2
 0 10 7 -10 -2 0 5
 0 10 0 -10 -2 7 5
 0 10 0 -10 -2 5 7
 0 10 0 -10 -2 5 7
```

# 3 Coding (15 Points Each)

1. Write a program that will use a random number generator to toss a coin as many times as it takes to get 20 heads in a row, and store all of the results in an arraylist. This number of tosses should be output to the screen. The program will then run through the arraylist and determine the longest run of consecutive tails and output that number to the screen.

---

**Solution:**

```
1  import java.util.ArrayList;
2
3  public class Exam3Prog1 {
4
5      public static void main(String[] args) {
6          ArrayList<Integer> tosses = new ArrayList<Integer>();
7
8          int headcount = 0;
9          while (headcount < 20) {
10             int toss = (int) (Math.random() * 2);
11             tosses.add(toss);
12             if (toss == 0)
13                 headcount++;
14             else
15                 headcount = 0;
16         }
17
18         System.out.println("Total number of tosses = " + tosses.size());
19
20         int tailcount = 0;
21         int maxtailcount = 0;
22         for (int i = 0; i < tosses.size(); i++){
23             if (tosses.get(i) == 1)
24                 tailcount++;
25             else{
26                 if (tailcount > maxtailcount)
27                     maxtailcount = tailcount;
28                 tailcount = 0;
29             }
30         }
31         System.out.println("Maximum run of consecutive tails = " + maxtailcount);
32     }
33 }
```

2. Write a program that will take the number of rows and columns as input, create a two-dimensional array of integers of that size, populate the array with random integers between 50 and 100 (inclusive), and then print it out. The program then creates an array of doubles of the same size, for each row of the integer array, the program finds the maximum entry in that row and then divides this value into each entry of the row, these values are to be stored in the double array. Finally, print out the double array. A program run would look like the following.

```
Input rows: 4
Input cols: 7

    52     73     98     84     58     99     74
    85     67     78     83     98     82     83
    98     98     79     92     81     64     82
    74     79     82     90    100     85     63

 0.525     0.737     0.990     0.848     0.586     1.000     0.747
 0.867     0.684     0.796     0.847     1.000     0.837     0.847
 1.000     1.000     0.806     0.939     0.827     0.653     0.837
 0.740     0.790     0.820     0.900     1.000     0.850     0.630
```

So in row 1, the maximum entry is 99, and $52/99 = 0.525$, $73/99 = 0.737$ and so on. For row 2, the maximum entry is 98, and $85/98 = 0.867$, $67/98 = 0.684$ and so on.

---

**Solution:**

```
1  import java.util.Scanner;
2
3  public class Exam3Prog2 {
4
5      public static void PrintArray(int A[][]) {
6          int dim1 = A.length;
7          int dim2 = A[0].length;
8
9          for (int i = 0; i < dim1; i++) {
10             for (int j = 0; j < dim2; j++) {
11                 System.out.printf("%7d", A[i][j]);
12             }
13             System.out.println();
14         }
15     }
16
17     public static void PrintArray(double A[][]) {
18         int dim1 = A.length;
19         int dim2 = A[0].length;
20
21         for (int i = 0; i < dim1; i++) {
22             for (int j = 0; j < dim2; j++) {
23                 System.out.printf("%10.3f", A[i][j
                       ]);
24             }
25             System.out.println();
26         }
27     }
28
29     public static void main(String[] args) {
30         Scanner keyboard = new Scanner(System.in);
31         System.out.print("Input rows: ");
32         int rows = keyboard.nextInt();
33         System.out.print("Input cols: ");
34         int cols = keyboard.nextInt();
35         int[][] A = new int[rows][cols];
36         double[][] B = new double[rows][cols];
37
38         for (int i = 0; i < rows; i++)
39             for (int j = 0; j < cols; j++) {
40                 A[i][j] = (int) (Math.random() *
                       51) + 50;
41             }
42
43         for (int i = 0; i < rows; i++) {
44             int rowmax = 0;
45             for (int j = 0; j < cols; j++) {
46                 if (A[i][j] > rowmax)
47                     rowmax = A[i][j];
48             }
49             for (int j = 0; j < cols; j++) {
50                 B[i][j] = (double) A[i][j] / rowmax
                       ;
51             }
52         }
53
54         System.out.println();
55         PrintArray(A);
56         System.out.println();
57         PrintArray(B);
58     }
59 }
```