1 Short Answer (10 Points Each)

1. For the following one-dimensional array, show the final array state after each pass of the three sorting algorithms. That is, after each iteration of the outside loop for each method.

   Solution:

   Bubble Sort:

   Original Array:

7	3	6	15	21	4	2

   ```
   Pass 1:  3  6  7 15  4  2 21
   Pass 2:  3  6  7  4  2 15 21
   Pass 3:  3  6  4  2  7 15 21
   Pass 4:  3  4  2  6  7 15 21
   Pass 5:  3  2  4  6  7 15 21
   Pass 6:  2  3  4  6  7 15 21
   ```

   Insertion Sort:

   Original Array:

7	3	6	15	21	4	2

   ```
   Pass 1:  3  7  6 15 21  4  2
   Pass 2:  3  6  7 15 21  4  2
   Pass 3:  3  6  7 15 21  4  2
   Pass 4:  3  6  7 15 21  4  2
   Pass 5:  3  4  6  7 15 21  2
   Pass 6:  2  3  4  6  7 15 21
   ```

   Selection Sort:

   Original Array:

7	3	6	15	21	4	2

   ```
   Pass 1:  7  3  6 15  2  4 21
   Pass 2:  7  3  6  4  2 15 21
   Pass 3:  2  3  6  4  7 15 21
   Pass 4:  2  3  4  6  7 15 21
   Pass 5:  2  3  4  6  7 15 21
   Pass 6:  2  3  4  6  7 15 21
   ```

2. Write a method that takes in a one-dimensional array of integers and an integer target value as its only parameters. The method is to do the binary search for the target value in the array. If the target value is found the method should return the index of the target value and if the element is not found then -1 should be returned by the method. Assume the array is already sorted in ascending order.

   Solution:

```java
static int binarySearch(int[] A, int N) {
    int lowestPossibleLoc = 0;
    int highestPossibleLoc = A.length - 1;
    while (highestPossibleLoc >= lowestPossibleLoc) {
        int middle = (lowestPossibleLoc + highestPossibleLoc) / 2;
        if (A[middle] == N) {
            return middle;
        } else if (A[middle] > N) {
            highestPossibleLoc = middle - 1;
        } else {
            lowestPossibleLoc = middle + 1;
        }
    }
    return -1;
}
```

3. Write a method that takes in a one-dimensional array of integers as its only parameter and sorts it using either the bubble sort, insertion sort, or selection sort.

   Solution:

```java
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;
            }
        }
    }
}
```

```
12
13 public static void insertionSort(int[] A) {
14     for (int itemsSorted = 1; itemsSorted < A.length; itemsSorted++) {
15         int temp = A[itemsSorted];
16         int loc = itemsSorted - 1;
17         while (loc >= 0 && A[loc] > temp) {
18             A[loc + 1] = A[loc];
19             loc = loc - 1;
20         }
21         A[loc + 1] = temp;
22     }
23 }
24
25 public static void selectionSort(int[] A) {
26     for (int lastPlace = A.length - 1; lastPlace > 0; lastPlace--) {
27         int maxLoc = 0;
28         for (int j = 1; j <= lastPlace; j++)
29             if (A[j] > A[maxLoc])
30                 maxLoc = j;
31
32         int temp = A[maxLoc];
33         A[maxLoc] = A[lastPlace];
34         A[lastPlace] = temp;
35     }
36 }
```

2 Program Trace (20 Points)

Write the output of the following program.

```
1  import java.util.Scanner;
2
3  public class Exam3Trace {
4
5      public static void main(String[] args) {
6          Scanner kb = new Scanner(System.in);
7
8          int A[] = new int[7];
9          A[0] = 7;
10         A[1] = 1;
11         A[2] = 3;
12         A[3] = 9;
13         A[4] = 2;
14         A[5] = 5;
15         A[6] = 4;
16
17         System.out.println(A[0] + " " + A[1] + " " + A[2] + " " + A[3] + " " + A[4] + " " + A[5] + " " + A[6]);
18
19         for (int i = 0; i < 6; i++) {
20             A[i + 1] += A[i];
21         }
22
23         System.out.println(A[0] + " " + A[1] + " " + A[2] + " " + A[3] + " " + A[4] + " " + A[5] + " " + A[6]);
24
25         for (int i = 0; i < 7; i++) {
26             if (i <= 1)
27                 A[i] = 1;
28             else {
29                 A[i] = A[i - 1] + A[i - 2];
30             }
31         }
32
33         System.out.println(A[0] + " " + A[1] + " " + A[2] + " " + A[3] + " " + A[4] + " " + A[5] + " " + A[6]);
34
35         for (int i = 0; i < 7; i++) {
36             A[i] = A[(5 * i) % 7];
37         }
38
39         System.out.println(A[0] + " " + A[1] + " " + A[2] + " " + A[3] + " " + A[4] + " " + A[5] + " " + A[6]);
40
41         A[0] = 6;
42         for (int i = 1; i < 7; i++) {
43             if (A[i - 1] % 2 == 0)
44                 A[i] = A[i - 1] / 2;
45             else
46                 A[i] = 3 * A[i - 1] + 1;
47         }
48
49         System.out.println(A[0] + " " + A[1] + " " + A[2] + " " + A[3] + " " + A[4] + " " + A[5] + " " + A[6]);
50     }
51 }
```

Solution:

```
7 1 3 9 2 5 4
7 8 11 20 22 27 31
1 1 2 3 5 8 13
1 8 3 8 13 13 3
6 3 10 5 16 8 4
```

3 Coding (30 Points Each)

1. Below is the main program and output of the program. You will be writing each of the methods that the program calls. The methods must work with the function calls made in the main without alteration. Each method has a short description on what it is supposed to do.

```
1  public static void main(String[] args) {
2      Scanner kb = new Scanner(System.in);
3
4      System.out.print("Input the array size: ");
5      int size = kb.nextInt();
6
7      int A[] = new int[size];
8
9      PopulateArray(A);
10     Print(A);
11     System.out.println();
12     System.out.println("Max = " + Max(A));
13     System.out.println("Min = " + Min(A));
14     System.out.println("Odds = " + NumbeOfOdds(A));
15     System.out.println("Exens = " + NumbeOfEvens(A));
16     System.out.println("Sum = " + Sum(A));
17 }
```

Program Output:

```
Input the array size: 10
246 219 391 689 -110 813 274 -960 -933 -660
Max = 813
Min = -960
Odds = 4
Exens = 6
Sum = -31
```

PopulateArray: Loads the array with random numbers between -1000 and 1000 inclusively.

Print: Prints the array to the screen with one space between each entry.

Max: Returns the maximum number being stored in the array.

Min: Returns the minimum number being stored in the array.

NumbeOfOdds: Returns the number of odd numbers in the array.

NumbeOfEvens: Returns the number of even numbers in the array.

Sum: Returns the sum of all the elements in the array.

Solution:

```
1  import java.util.Scanner;
2
3  public class Exam3Prog1 {
4
5      public static void main(String[] args) {
6          Scanner kb = new Scanner(System.in);
7
8          System.out.print("Input the array size: ");
9          int size = kb.nextInt();
10
11         int A[] = new int[size];
12
13         PopulateArray(A);
14         Print(A);
15         System.out.println();
16         System.out.println("Max = " + Max(A));
17         System.out.println("Min = " + Min(A));
18         System.out.println("Odds = " + NumbeOfOdds(A));
19         System.out.println("Exens = " + NumbeOfEvens(A));
20         System.out.println("Sum = " + Sum(A));
21     }
22
23     public static void Print(int A[]) {
24         for (int i = 0; i < A.length; i++) {
```

```
25              System.out.print(A[i] + " ");
26          }
27      }
28
29      public static void PopulateArray(int A[]) {
30          for (int i = 0; i < A.length; i++) {
31              A[i] = (int) (Math.random() * 2001) - 1000;
32          }
33      }
34
35      public static int Max(int A[]) {
36          int max = A[0];
37
38          for (int i = 0; i < A.length; i++)
39              if (max < A[i])
40                  max = A[i];
41
42          return max;
43      }
44
45      public static int Min(int A[]) {
46          int min = A[0];
47
48          for (int i = 0; i < A.length; i++)
49              if (min > A[i])
50                  min = A[i];
51
52          return min;
53      }
54
55      public static int NumbeOfOdds(int A[]) {
56          int num = 0;
57
58          for (int i = 0; i < A.length; i++)
59              if (A[i] % 2 == 1)
60                  num++;
61
62          return num;
63      }
64
65      public static int NumbeOfEvens(int A[]) {
66          return A.length - NumbeOfOdds(A);
67      }
68
69      public static int Sum(int A[]) {
70          int sum = 0;
71
72          for (int i = 0; i < A.length; i++)
73              sum += A[i];
74
75          return sum;
76      }
77
78  }
```

2. Write the code to create an object named `Triangle` that will store the lengths of the three sides of the triangle, decimal numbers of course. The class should have a constructor that takes in the three sides as parameters and sets the data member values of the three sides to the parameter values. The class should have accessor methods for returning the lengths of each side, a method `Perimeter` that returns the perimeter of the triangle, a method `Area` that returns the area of the triangle, a method `isRight` that returns true if the triangle is a right triangle and false if not, and a method `isTriangle` that returns true if the lengths of the sides make a triangle and false if not. You do not need to do error checking on the lengths of the sides, and recall that the area of a triangle can be calculated by,

$$A = \sqrt{p(p-a)(p-b)(p-c)}$$

where p is one half of the perimeter and a, b, and c represent the lengths of the sides of the triangle.

Solution:

```
1  public class Triangle {
2      private double a;
3      private double b;
4      private double c;
5
6      public Triangle(double s1, double s2, double s3) {
7          a = s1;
8          b = s2;
9          c = s3;
10     }
11
12     public double getSide1() {
13         return a;
14     }
15
16     public double getSide2() {
17         return b;
18     }
19
20     public double getSide3() {
21         return c;
22     }
23
24     public double Area() {
25         double p = (a + b + c) / 2;
26         return Math.sqrt(p * (p - a) * (p - b) * (p - c));
27     }
28
29     public double Perimeter() {
30         return a + b + c;
31     }
32
33     public boolean isRight() {
34         boolean righttri = false;
35         if (a * a + b * b == c * c)
36             righttri = true;
37
38         if (a * a + c * c == b * b)
39             righttri = true;
40
41         if (c * c + b * b == a * a)
42             righttri = true;
43
44         return righttri;
45     }
46
47     public boolean isTriangle() {
48         boolean tri = true;
49
50         double longleg = a;
51         if (b > longleg)
52             longleg = b;
53         if (c > longleg)
54             longleg = c;
55
56         if (a + b + c - longleg <= longleg)
57             tri = false;
58
59         return tri;
60     }
61 }
```