1. (12 points) Trace the integer variables in the following code:

```
int x = 5;
int y = 10;
do{
        x++; // 6, 9, 10
        y--; // 9, 5, 4
        while (y >8){ // 9 > 8, 5 > 8, 4 > 8
                x = x+2; // x = 6 + 2 = 8
                y = y-3; // y = 9 - 3 = 6
        }
} while (x < 10); // 8 < 10, 9 < 10, 10 < 10
```

x	5	6	8	9	10				
y	10	9	6	5	4				

b.

```
for (int i=0; i<5; i=i+2){
        for (int j=i; j>0; j--){ }
}
```

i	0	2	2	2	4	4	4	4	4
j	0	2	1	0	4	3	2	1	0

2. (12 points) An integer variable is assigned a value representing inches. Compute and print how many miles, yards, feet and inches that number represents. The rules are: 1 foot equal 12 inches; 36 inches equal 1 yard; 5280 feet equals 1 mile. For example,
245897 inches equals 3 miles 1550 yards 1 foot and 5 inches

```
public static void main(String []args){
        Scanner sc = new Scanner (System.in());
        int totalInches = sc.nextInt(); //245897
        final int inchesInFoot = 12;
        final int inchesInYard = 36;
        final int feetInMile = 5280;
        // compute # of inches in a mile
        final int inchesInMile = feetInMile * inchesInFoot;
        int miles = totalInches / inchesInMile;
        int remainingInches = totalInches % inchesInMile;
        int yards = remainingInches / inchesInYard;
        remainingInches = remainingInches % inchesInYard;
        int feet = remainingInches / inchesInFoot;
        remainingInches = remainingInches % inchesInFoot;
        System.out.println (miles + " miles " + yards + " yards " + feet + " feet
" + remainingInches + " inches");
}
```

3. (6 points) SHOW THE WORK you used to perform the following conversions:

a. 1100110 (base 2) to base 10 Answer _________102_________

1100110 >>> 7 digits >>> base 2^6 - 2^0

(1 * 2^6) + (1 * 2^5) + (0 * 2^4) + (0 * 2^3) + (1 * 2^2) + (1 * 2^1) + (0 * 2^0)

64 + 32 + 0 + 0 + 4 + 2 + 0 = 102

b. AB (base 16) to base 10 Answer _______171_______

AB >>> 2 digits >> base 16^1 - 16^0

A = 10, B = 11

(10 * 16^1) + (11 * 16^0)

160 + 11 = 171

c. 65 (base 10) to base 2 Answer _____1000001______

65 / 2 = 32	remainder = 1
32 / 2 = 16	remainder = 0
16 / 2 = 8	remainder = 0
8 / 2 = 4	remainder = 0
4 / 2 = 2	remainder = 0
2 / 2 = 1	remainder = 0
1 / 2 = 0	remainder = 1

1 0 0 0 0 0 1

4. (16 points) For each part below, fill in the table to trace the values as they are passed from main and changed in the method.

a.

```
int [ ] array = {1,2,3,4,5};
int x = 12;
int y = doOne(array,x); // method call 1

int [ ] array2 = {1,2,3,4,5};
int w = doOne(array2, array2[3]); // method call 2

} // end of main

public static int doOne(int[] arr, int x){
        for (int i=0; i<arr.length-1; i++)
                arr[i] = arr[i]*arr[i+1];
                // arr[0] = arr[0] * arr[0+1] = 1 * arr[1] = 1 * 2 = 2
                // arr[1] = arr[1] * arr[1+1] = 2 * arr[2] = 2 * 3 = 6
                // arr[2] = arr[2] * arr[2+1] = 3 * arr[3] = 3 * 4 = 12
                // arr[3] = arr[3] * arr[3+1] = 4 * arr[4] = 4 * 5 = 20
                // arr[0]= arr[0] * arr[0+1] = 1 * arr[1] = 1 * 2 = 2
                // arr[1] = arr[1] * arr[1+1] = 2 * arr[2] = 2 * 3 = 6
                // arr[2] = arr[2] * arr[2+1] = 3 * arr[3] = 3 * 4 = 12
                // arr[3] = arr[3] * arr[3+1] = 4 * arr[4] = 4 * 5 = 20

        x = x+3; // x = 12 + 3 = 15  // x = 4 + 3 = 7
        return x; // return 15  // return 7
}
```

x in main before call to doOne	12
array in main before call to doOne	1,2,3,4,5
arr in doOne before loop	1,2,3,4,5
x in doOne before loop	12
arr in doOne after loop	2,6,12,20,5
x in doOne after loop	15
x in main after call to doOne	12
y in main after call to doOne	15
array in main after call to doOne	2,6,12,20,5

array2 in main before call to doOne	1,2,3,4,5
arr in doOne before loop	1,2,3,4,5
x in doOne before loop	4
arr in doOne after loop	2,6,12,20,5
x in doOne after loop	7
w in main after call to doOne	7
array2 in main after call to doOne	2,6,12,20,5

```
b.      int [] array3 = {1,2,3,4,5};
        int s = 12;
        int v = 32;
        int [] num = doTwo(array3, v,s);

 } // end of main

 public static int[] doTwo(int[] arr, int x, int y){

        int[] ret = new int[2];
        x = arr[2] ; // x = 3
        y = arr[3] ; // y = 4
        ret[0] = x+arr[1]; // ret[0] = 3 + 2 = 5
        ret[1] = y-arr[0]; // ret[1] = 4 - 1 = 3
        return ret; // {5,3}
 }
```

array3 before call to doTwo	1,2,3,4,5
s before call to doTwo	12
v before call to doTwo	32
arr in doTwo	1,2,3,4,5
x in doTwo	3
y in doTwo	4
ret in doTwo	5,3
array3 in main after call to doTwo	1,2,3,4,5
num in main after call to doTwo	5,3
s in main after call to doTwo	12
v in main after call to doTwo:	32

5. (12 points)

a. Given 6 integers i,j,k,l,m and n all initialized and a method max3 which returns the largest of any 3 numbers, write one Java statement to find the maximum of the 6 integers using any combination of Math.max and max3

```
Math.max(max3(i,j,k), max(l,m,n));
//finds the maximum out of i,j,k
//and the maximum out of l,m,n
//and then the maximum between the two
```

b. Write a Java expression that would compute the following:

$$\frac{x^2 - \frac{1}{(y+3)}}{\frac{1}{z} + \sqrt{x}}$$

(Math.pow(x,2) - (1/(y+3)))/((1/z)+Math.sqrt(x));

c. Perform a binary search on the following array searching for the number 18

int[] nums = {12, 77, 87, 89, 100, 117, 125, 189, 235, 529, 1000};

List the high, low and midpoint values at each step until the algorithm stops. DO NOT write any code. Enter into the table below the low, mid and high values as the binary search progresses. You may use either the subscript (index) values or the actual number stored at the subscript. Not all rows of the table may be needed.

Low	Mid	High
12 (0)	117 (0 + (10-0)/2)	1000 (10)
12 (0)	87 (0 + (4-0)/2)	100(4)
12 (0)	12 (0 + (1-0)/2)	77 (1)
77 (1)	77 (1 + (1-0)/2)	77 (1)

6. (12 points) You are given a string containing a series of 9 digit zip codes with a dash separating the two parts and one space separating each full zip code from the next one. For example,

String str="11230-1234 11011-3489 07621-8845";

Write Java code to print the first 5 digits of each zip code followed by the last 4 digits of the zip code as a separate value.

For example:

```
11230 1234
11011 3489
07621 8845
```

NOTE: The first part of the full zip code is always 5 digits, the second part is always 4 digits and there's always a dash in between the two parts.

Keep in mind that your code must handle a String that has any number of such pairs, not just the three in the example above

```
import java.util.Scanner; //Scanner to read String
public class Question6 {
    public static void main (String [] args) {
        // CAN create Scanner to read str from input BUT not necessary
        String str = "11230-1234 11011-3489 07621-8845"; //TBD ??
        Scanner sc = new Scanner (str); //read from String
        while(sc.hasNext()) { //read until end of str
            //read individual zipCode String
            String zipCode = sc.next(); //11230-1234
            int dash = zipCode.indexOf('-'); //find location of '-'
            //substring
            String first = zipCode.substring(0,dash); //first five
            String second = zipCode.substring(dash+1); //last four
            System.out.println(first + " " + second); //done last after zip
code is split
        }

    }
}
```

7. (30 points) Write a complete Java program, including at least one comment in the main program and one in each method, to do the following:

The program will read in an unknown number of records from a file. Assume you won't have more than 100 records in total, though the actual number of records can be less than 100. Each record contains a sales rep's first name and miles traveled on two trips per year. For example, *Pauline 167.8 567.0*

Main:

- **Declare an output file to be used in main and passed to one or more methods from main. All output from main and methods will be printed to this file.**
- **Invoke the first method (below) to read the data from the input file, compute averages and store information in the arrays. The method returns the number of records read in.**
- **Next, invoke the second method (below) which computes the average length of all trips and then prints to the output file how many individual averages are above, equal to and below the overall average.**
- **Finally, invoke the third method (below) to sort the name and individual average trip length arrays in parallel**
- **In main, print to the output file each sales rep name and average trip length. The names and average trip length should print in columns, right adjusted. All trip lengths should be printed with two decimal places.**

Methods:

1. **This method has two parameters: a String array for names and a double array for the average trip length per individual sales rep. It returns an integer**
   a. **Declare the input file, read the records in the file and store each first name in an array (assume there are no duplicate names) and the average number of miles travelled for that sales rep in another array. You do not need to store the individual trip length.**
   b. **Return the number of sales rep records read in.**

2. **This method has three parameters – the output file, double array of individual average trip length and an integer representing the total number of records read in.**
   a. **Compute the overall average trip length across all sales reps.**
   b. **Compute and print to the output file how many sales reps' individual average trip length were above the overall average trip length, below the overall average trip length and equal to the overall average trip length.**

3. **This method has three parameters – String array of names, double array of individual average trip length and an integer representing the total number of records read in by the first method.**
   a. **Sort the sales rep names in descending (reverse) alphabetical order synchronizing the individual sales rep's average trip length in the parallel array.**

```
import java.util.*;
import java.io.*;

public class Question7{

    public static void main(String []args){
        PrintWriter output = new Printwriter ("output.txt");
        final int CAPACITY = 100;
        String [] names = new String [CAPACITY];
        double [] averageTripLength = new double [CAPACITY];
        int size = readData(names, averageTripLength);
        average(output, averageTripLength, size);
        sort(names, averageTripLength, size);

        for(int i = 0; i < size; i++){
            output.printf(“%10s\t%10.2f\n”, names[i],
    averageTripLength[i]);
        }

        output.close();

    }

    //method 1
    public static int readData (String [] names, double []
averageTripLength){
        File input = new File("input.txt");
        Scanner sc = new Scanner(input);
        int count = 0;
        while(sc.hasNext()){
            names[count] = sc.next();
             double trip1 = sc.nextDouble();
             double trip2 = sc.nextDouble();
             averageTripLength[count] = (trip1 + trip2)/2;
             count++;
         }
         return count;
    }

    //method 2
    public static void average (PrintWriter output, double []
averageTripLength, int size){
        double sum = 0;
        for(int i = 0; i < size; i++){
            sum += averageTripLength[i];
        }

        double overallAverage = sum/size;
        int above = 0, below = 0, equalTo = 0;

        for(int j = 0l j < size; j++){
            if(averageTripLength[i] > overallAverage){
```

```
                                above++;
                            }
                           else if (averageTripLength[i] < overallAverage){
                                below++;
                                }
                           else{
                                equalTo++;
                           }
                    }

                    output.println("The overall average trip length: " +
                    overallAverage);
                    output.println("The number of sales reps individual average
                    trip lengths above the overall average trip length: " + above);
                    output.println("The number of sales reps individual average
                    trip lengths below the overall average trip length: " + below);
                    output.println("The number of sales reps individual average
                    trip lengths equal to the overall average trip length: " +
                    equalTo);
              }

              //method 3
              public static void sort (String [] names, double [] averageTripLength,
        int size){
                    String temp;
                    double temp2 = 0;

                    // Sorting strings using bubble sort
                    for (int i = 0; i < size-1; i++){
                          for (int j = 0; j < size-i-1; j++){
                          if (names[j].compareTo(names[j+1]) > 0){ //descending
        order
                                temp = names[j];
                                names[j] = names[j+1];
                                names[j+1] = temp;

                                temp2 = averageTripLength[j];
                                averageTripLength[j] = averageTripLength[j+1];
                                averageTripLength[j + 1] = temp2;
                          }
                      }
                    }
                 }
        }
```