Problem 1: Loops (10 points)

a. Consider the Q1a program. Describe, in one sentence, what the Q1a does.

```
public class Q1a
{
    public static void main (String[] args)
    {
    int n = 1000;
    int mySum = 0;

    for (int i = n; i >= 0; i--)
    {
        if (i % 2 == 0)
        {
            mySum += i;
        }
    }
    StdOut.println(mySum);
    }
}
```

Loop adds up even numbers from 0 to 1000 (3 points)

b. Complete the program Q1b that reads a set of integer values into the integer array numbers (already done for you). Then it adds up the positive values in the numbers array until it reads the item 999 (999 is not included in the sum). At that point it prints the sum of those numbers and their average.

Example 1
java Q1b 5 -3 8 999 7 1
10
5

Example 2
java Q1b 5 -3 8 7 1
18
5.25

```
public class Q1b
{
    public static void main (String[] args)
    {
     int[] numbers = new int[args.length];
     for (int i = 0; i < args.length; i++) {
        numbers[i] = Integer.parseInt(args[0]);
     }

    // INSERT YOUR CODE HERE
     int    mySum = 0;
     double myAve = 0;
     int  myCount = 0;
     for (int i = 0; i < numbers.length; i++) { // (3 points)
         if (Numbers[i] == 999) {               // (3 points)
             break;
```

```
            } else{
                if (Numbers[i] > 0) {.      // (2 points)
                    mySum += Numbers[i];
                    myCount += 1;
                }
            }
        }
        myAve = (double)mySum/(double)myCount; // (1 point)
        StdOut.println(mySum + " " + myAve);    // (1 point)
    }
}
```

Problem 2: Arrays (15 points)

Complete the method `endsInFive(int[][] array, int c)` that takes a 2D-array of integers and an integer value as arguments. endsInFive returns the number of values in column c that end in 5. It is possible that array is ragged, meaning that the number of columns in each row is different.

For example, 2 is returned if the following 2D-array is passed as the first argument of endsInFive and 3 is the second argument. The value 25 and 115 end in five in column 3.

234	34	13	25	
762	2	995		
125	345	54	859	
21	467	15	115	785

```
public static int endsInFive(int[][] array, int c) {

  // WRITE YOUR CODE HERE

  int count = 0;
  for ( int r = 0; r < array.length; r++ ) { // looping over rows (5pts)

     if ( c < array[r].length ) {            // check column exists (3pts)
       if ( array[r][c] % 5 == 0 ) {   // check if number ends in 5 (3pts)
         count += 1;                         // increment count (2pts)
       }
     }
  }
  return count;                                 // returns count (2pts)
}
```

Problem 3: Functions (25 points)
For each of the following three problems, choose the correct output from the box on the right. (4 points each)

1)

```
public class Q3a {
    public static void main(String[] args){
        var a = " coming!";
        System.out.println(methodC(methodB(a)));
    }
    public static String methodF(String d){
        return "Thanksgiving " + d;
    }
    public static String methodC(String g){
        return methodF(g);
    }
    public static String methodB(String c){
        String a = "of ";
        return a + "Turkeys  ";
    }
}
```

a) Turkeys of Thanksgiving coming!
b) Thanksgiving coming!
c) Thanksgiving of Turkeys
d) None of the above.

2)

```
public class Q3b {
    public static void main(String[] args){
        var a = "coming!";
        System.out.println(methodC(methodB(a, a), a));
    }

    public static String methodF(String d, String a){
        return "Thanksgiving " + a;
    }
    public static String methodC(String g, String a){
        return methodF(g,a);
    }

    public static String methodB(String c, String b){
        String a = "of ";
        return "Turkeys  " + a;
    }
}
```

a) Turkeys of Thanksgiving coming!
b) Thanksgiving coming!
c) Thanksgiving of Turkeys
d) None of the above.

3)

```
public class Q3c {
    public static void main(String[] args){
        var a = " coming!";
        System.out.println(methodB(methodF(a)));
    }

    public static String methodF(String d){
        return " of Thanksgiving" + d;
    }
    public static String methodC(String g){
        return methodF(g);
    }

    public static String methodB(String c){
        String a = "of ";
        return "Turkeys" + c;
    }
}
```

a) Turkeys of Thanksgiving coming!
b) Thanksgiving coming!
c) Thanksgiving of Turkeys
d) None of the above.

Solution (4 points each)
java Q3a
Thanksgiving of Turkeys
$ java Q3b
Thanksgiving coming!
$ java Q3c
Turkeys of Thanksgiving coming!

#2. Two numbers, num1 and num2, are said to be "*friendly*" if the sum of proper factors of num1 is equal to num2 and the sum of the proper factors of num2 is equal to num1.

The method **main** below is to print either "**Friendly**" or "**Not Friendly**" to indicate whether two integer command line inputs, `num1` and `num2`, are "*friendly*". Complete **main** so that is works as intended. You **must** call method `sumOfDivisors` in your solution.

You may assume that the method `sumOfDivisors` returns the sum of the proper factors of its integer parameter.

```
public class Friendly {
    public static void main(String[] args){
         // your code goes here
    }

    /*
    This method returns the sum of the proper divisors of n.
    A proper divisor is a number between 1 and n-1 inclusive that
    Divides into n evenly. YOU DO. NOT HAVE TO WRITE THIS CODE.
    Assume this method works as intended.
    */
    public static int sumOfDivisors(int n){
      //code not shown—do not write this code
      }
}
```

Solution (13 pts)

```
public static void main(String[] args){

        int num1 = Integer.parseInt(args[0]);
        int num2 = Integer.parseInt(args[1]);
        int firstSum = sumOfDivisors(num1);
        int secondSum = sumOfDivisors(num2);
        if (firstSum == num2 && secondSum == num1){
            System.out.println("Friendly");
        }
        else{
            System.out.println("Not Friendly");
        }
    }
```

3 pt for command line input

4 pts for TWO calls to sumOfDivisors with correct arguments

4 pts for correct comparisons (must check both) and must be referring to results of sumOfDivisors calls

2 pts for correct output

Note: lose 8 pts if no calls to sumOfDivisors.

Problem 4: Recursion (25 points)

Part 1 (15 points)

Consider the following recursive function foo.

```
1   int foo(int a, int b) {
2     if (a < b)
3        return 0;
4    else
5       return 1 + foo(a-b, b);
6  }
```

1.A (2 pts) Which line(s) handles the base case?

Lines 2 and 3

1.B (3 pts) What is the outcome if foo(0,0) is called?

Non-terminating process

1.C (2 points) What are the values returned for foo(4,2) and foo(5,3) respectively?

2 and 1

1.D (8 points) Briefly (1-2 lines) explain the purpose of the function foo

returns the integer quotient of **a** divided by **b**

Part 2 (10 points)
Assume that the function drawSquare(double x, double y, double halfLen) draws a square of the specified size, centered at (*x*, *y*). For example, drawSquare(0,0,5) draws a square centered at (0,0) with each side of length 10.

Consider the following function foo defined as

```
public void foo(double x, double y, double len, int n) {
      if (n<1) return;
      drawSquare (x, y, len);
      foo(x-len/2, y-len/2, len/2,n-1);
      foo(x-len/2, y+len/2, len/2,n-1);
      foo(x+len/2, y+len/2, len/2,n-1);
      foo(x+len/2, y-len/2, len/2,n-1);
}
```

2.A (1 pts) How many squares are drawn if foo(0,0,5,1) is called? 1 = 4^0

2.B (2 pts) How many squares are drawn if foo(0,0,5,2) is called?
5 = 4^0 + 4^1

2.C (2 pts) How many squares will be drawn if foo(0,0,5,3) is called?
21 = 4^2 + 4^1+ 4^0

2.D. (3 pts) How many squares will be drawn if foo(0,0,5,n) is called? The answer should be given as a function of n
4^(n-1) + … 4^0 = (4^n – 1)/(4-1)

Problem 5: Object-Oriented Programming (25 points)

Suppose we have a 1-D array of Strings such that:

- Each string records individual scores for different quizzes for one student.
- Each string has the form "d1;d2;d3;..;dn;" where d1 to dn are all double values and they are separated using the character ';'. Note that there is one ';' after the last double value.
- Each string contains the same number of double values.

Write a method that takes a 1-D array of such Strings and an integer n as inputs, where n means the number of double values in each String in the array and returns a 2-D array containing double values. In the two 2-D array, each row records scores of one student and each column contains scores of one exam for all students.

For example, if the parameters are {"1;2;3;4;", "5;6;7;8;", "9;10;11;12;"} and 4, then the output would be a 3*4 2-D array which has the value {{1.0, 2.0, 3.0, 4.0}, {5.0, 6.0, 7.0, 8.0}, {9.0, 10.0, 11.0, 12.0}}. You are supposed to use the provided String library methods to operate these string values. Here is the documentation for String:
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/String.html

The method signature is given below:
public static double[][] convert(String[] exams, int n)

Sol:

```
public static double[][] convert(String[] exams, int n){
     double[][] ret = new double[exams.length][n];
     for(int i=0; i< exams.length; i++) {
          String exam = exams[i];
          int index = exam.indexOf(";");
          int j = 0;
          while(index != -1) {
               double score =
Double.parseDouble(exam.substring(0, index));
               ret[i][j] = score;
               j++;
               exam = exam.substring(index+1);
               index = exam.indexOf(";");
          }
     }
     return ret;
}
```

Grading:
5 pts for creating a 2-D array with exams.length and n where exams.length and n are both worth 1 pt.

4 pts for a loop to handle each element in the parameter exams.

4 pts for a loop to handle each score in a string variable.

3 pts for separating the scores, the sample solution uses indexOf, if the students use charAt to check each character, it is also ok. There might be some other solutions too. But must use methods in the String class. Otherwise, no points.

2 pts for getting one score, the sample solution uses substring, if the students use + to compose one score, it is also ok. There might be some other solutions too. But must use methods in the String class. Otherwise, no points.

2 pts for using Double.parseDouble to convert String to double.

2 pts for correctly setting a 2-D cell with a score.

2 pts for returning a 2-d array value.