Competitive Programming iC-PROM 2017

CONTEST (Solution)

This problem set contains 10 questions (A – J)

26 November 2017

Organized by

Computer Science Department, UiTM Perlis

PROBLEM A	**THE NEXT NUMBER**

A relatively simple task for humans to do is to identify the next number in an arithmetic sequence of numbers. For example, given the sequence of number 3, 6, and 9, we know that the next number in sequence is 12, since each number is followed by a number increasing it by 3.

Your task is to write a program that will deduce the next integer in a given sequence of three integers where the same arithmetic operation is applied to each integer to produce the next. The only arithmetic operations that will be used are addition, subtraction, multiplication, or division.

Input

The first line of the input contains an integer N ($1 \leq N \leq 5$), the number of test cases. Following the first line are the test cases. Each line of a test case contains three integers with each pair of integers separated by a single space.

The input must be read from standard input.

Output

The output will print the next integer number in the corresponding input sequence. The output is to be formatted exactly like that for the sample output given below.

Sample Input	Sample Output

`3`
`5 10 15`
`-10 20 -40`
`64 16 4`
	
	`20`
	`80`
	`1`

PROBLEM A	THE NEXT NUMBER

```
import java.util.*;

public class S1{
      public static void main(String[] args){
Scanner scan = new Scanner(System.in);
            Scanner scanN = new Scanner(System.in);

            int N = scanN.nextInt();
            String[] data = new String[N];

            for (int i=0; i<N; i++)
            data[i] = scan.nextLine();

            for(int i=0; i<N; i++)
            {
                  StringTokenizer st = new StringTokenizer(data[i]);
      double x = Double.parseDouble(st.nextToken());
                  double y = Double.parseDouble(st.nextToken());
                  double z = Double.parseDouble(st.nextToken());

                  if (y-x == z-y)
                        System.out.println((int)(z+y-x));
                  else
                        System.out.println((int)(z/(x/y)));
            }
      }
}
```

PROBLEM B — VOWEL FORMATION EXTRACTOR

A sentence consists of vowels, consonants, special characters and numbers. Imagine if you are able to count manually how many vowels of `a`, `e`, `i`, `o` and `u` from a sentence or strings.

The vowel extractor codes are able to extract vowels from a sentence or string and count number of occurrences of each vowel. The formation of vowels will be printed according to number of occurrences of each vowel.

Input

The first line of input contains an integer number N (1 ≤ N ≤ 5) represents the number of test cases. Following the first line are the test cases. Each line of a test case contains a sentence.

The input must be read from standard input.

Output

The output of the program should display as shown in the following sample of output.

Display the sentence of each test case before the formation of each vowel is printed.

The output must be written to standard output.

Sample Input

```
3
I love programming.
Why Dr Ng Cry?
Hmmmm... what happened if monkey
take over the OS class?
```

Sample Output

```
I love programming.
a
e
i
I
oo

Why Dr Ng Cry?

Hmmmm... what happened if monkey
take over the OS class?
aaaa
eeeeee
i
oo
O
```

PROBLEM B — VOWEL FORMATION EXTRACTOR

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

public class VowelFormationExtractor{
public static void main(String[] args){
        Scanner scan = new Scanner(System.in);
        String lineSeparator = System.getProperty("line.separator");
        scan.useDelimiter(lineSeparator);

        final int totSets = 3;

        String[] sentences = new String[3];

        System.out.println("\n Enter "+totSets+" sets of sentences : ");

        for (int i=0;i<totSets;i++) {
                sentences[i] = scan.next();
        }

        for (int i=0;i<totSets;i++) {
                int len = sentences[i].length();
                System.out.print("\n\n"+sentences[i]);
                int cnta=0, cntA=0, cnte=0, cntE=0, cnti=0, cntI=0, cnto=0, cntO=0, cntu=0, cntU=0;
                for (int x=0;x<len;x++)
                {
                        if (sentences[i].charAt(x) == 'a')
                                cnta++;

                        if (sentences[i].charAt(x) == 'A')
                                cntA++;

                        if (sentences[i].charAt(x) == 'e')
                                cnte++;

                        if (sentences[i].charAt(x) == 'E')
                                cntE++;

                        if (sentences[i].charAt(x) == 'i')
                                cnti++;

                        if (sentences[i].charAt(x) == 'I')
                                cntI++;

                        if (sentences[i].charAt(x) == 'O')
                                cntO++;

                        if (sentences[i].charAt(x) == 'o')
                                cnto++;

                        if (sentences[i].charAt(x) == 'u')
                                cntu++;

                        if (sentences[i].charAt(x) == 'U')
                                cntU++;
                }

                if (cnta > 0){
                        System.out.println();
                        for (int j=0;j<cnta;j++)
                                System.out.print("a");
                }
                if (cntA > 0){
                        System.out.println();
                        for (int j=0;j<cntA;j++)
                                System.out.print("A");
                }

                if (cnte > 0){
                        System.out.println();
                        for (int j=0;j<cnte;j++)
                                System.out.print("e");
                }

                if (cntE > 0){
                        System.out.println();
                        for (int j=0;j<cntE;j++)
                                System.out.print("E");
                }

                if (cnti > 0){
                        System.out.println();
                        for (int j=0;j<cnti;j++)
                                System.out.print("i");
                }


                if (cntI > 0){
                        System.out.println();
                        for (int j=0;j<cntI;j++)
                                System.out.print("I");
                }

                if (cnto > 0){
                        System.out.println();
                        for (int j=0;j<cnto;j++)
                                System.out.print("o");
                }

                if (cntO > 0){
                        System.out.println();
                        for (int j=0;j<cntO;j++)
                                System.out.print("O");
                }

                if (cntu > 0){
                        System.out.println();
                        for (int j=0;j<cntu;j++)
                                System.out.print("u");
                }

                if (cntU > 0){
                        System.out.println();
                        for (int j=0;j<cntU;j++)
                                System.out.print("U");
                }

        }

                System.out.println("\n\n\n\n\n");
        }
}
```

PROBLEM C	BINARY NUMBER CONVERSION

A binary number is a number expressed in the binary numeral system or base-2 numeral system which represents numeric values using two different symbols: typically 0 (zero) and 1 (one). For example the machine code is consists of binary number or hexadecimal instructions that a computer can respond directly.

The idea of this problem is to convert a binary number to the non-negative integer number.

Input

The first line of input contains an integer number N (1 ≤ N ≤ 5) represents the number of test cases. Following the first line are the test cases. Each line of test case contains a binary number. The length of each test case or the binary number is less than 6 digits.

The input must be read from standard input.

Output

The output of the program should display as shown in the following sample of output.

Display the integer number for each binary number. If the test case is not a binary number, display "`Invalid binary number`"

The output must be written to standard output.

Sample Input	Sample Output
5	
101011	43
11111	31
1110	14
1012	Invalid Binary Number
11	3

PROBLEM C	BINARY NUMBER CONVERSION

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

public class BinaryNumber{
public static void main(String[] args)
{
	Scanner scan = new Scanner(System.in);
	String lineSeparator = System.getProperty("line.separator");
	scan.useDelimiter(lineSeparator);

	final int totSets = 5;
	String[] binaryNumber = new String[totSets];
	int number = 0;

	System.out.println("\n Enter "+totSets+" binary numbers : ");

	for (int i=0;i<totSets;i++) {
		binaryNumber[i] = scan.next();
		while (binaryNumber[i].length() > 6)
		{
			System.out.println("\n Binary number exceeds 6 digits ");
			binaryNumber[i] = scan.next();
		}
	}

	for (int i=0;i<totSets;i++) {

		int len = binaryNumber[i].length();
		boolean yes = false;
		int cnt = 0;

		for (int x=0;x<len;x++){
			char digit = binaryNumber[i].charAt(x);
			if (digit == '1' || digit == '0')
				cnt++;
		}

		if (cnt == len)
			yes = true;
		else
			System.out.println("\n Invalid Binary Number ");
		if (yes == true)
		{
			if (len == 6)
			{
				if (binaryNumber[i].substring(0,1).equals("1"))
					number = number + 32;

				if (binaryNumber[i].substring(1,2).equals("1"))
```

```
                        number = number + 16;

                    if (binaryNumber[i].substring(2,3).equals("1"))
                        number = number + 8;

                    if (binaryNumber[i].substring(3,4).equals("1"))
                        number = number + 4;

                    if (binaryNumber[i].substring(4,5).equals("1"))
                        number = number + 2;

                    if (binaryNumber[i].substring(5,6).equals("1"))
                        number = number + 1;
                }

                else if (len == 5)
                {
                    if (binaryNumber[i].substring(0,1).equals("1"))
                        number = number + 16;

                    if (binaryNumber[i].substring(1,2).equals("1"))
                        number = number + 8;

                    if (binaryNumber[i].substring(2,3).equals("1"))
                        number = number + 4;

                    if (binaryNumber[i].substring(3,4).equals("1"))
                        number = number + 2;

                    if (binaryNumber[i].substring(4,5).equals("1"))
                        number = number + 1;
                }

                else if (len == 4)
                {
                    if (binaryNumber[i].substring(0,1).equals("1"))
                        number = number + 8;

                    if (binaryNumber[i].substring(1,2).equals("1"))
                        number = number + 4;

                    if (binaryNumber[i].substring(2,3).equals("1"))
                        number = number + 2;

                    if (binaryNumber[i].substring(3,4).equals("1"))
                        number = number + 1;
                }

                else if (len == 3)
                {
                    if (binaryNumber[i].substring(0,1).equals("1"))
                        number = number + 4;

                    if (binaryNumber[i].substring(1,2).equals("1"))
                        number = number + 2;

                    if (binaryNumber[i].substring(2,3).equals("1"))
                        number = number + 1;
                }
```

```
                    else if (len == 2)
                    {
                          if (binaryNumber[i].substring(0,1).equals("1"))
                                number = number + 2;

                          if (binaryNumber[i].substring(1,2).equals("1"))
                                number = number + 1;
                    }

                    else if (len == 1)
                    {
                          if (binaryNumber[i].substring(0,1).equals("1"))
                                number = number + 1;
                    }
                    System.out.println("\n"+number);
                    number = 0;
              }

              cnt=0;
              yes=false;

        }

}
}
```

PROBLEM D	**RACING CAR**

Write a program that is able to calculate the number of lamp posts passed by a car when it is driving at specified speed for a certain period of time. Each lamp post is placed 35m from each other.

Input

1. Average travelling speed of the car.
2. Amount of time in hours.

Output

Number of lamp post.

Sample input	Sample output
110 4	1257
90 8	2057

PROBLEM D	**RACING CAR**

```
#include<iostream>
using namespace std;

int main()
{
    int speed;
    float time;

    cout<<"What is your average travelling speed? ";
    cin>>speed;
    cout<<"How long you have been travelling (in hours)?  ";
    cin>>time;

    float distance = time * speed;
    float distance_in_m = distance * 100;
    int no_lampPost = distance_in_m / 35;

    cout<<"\nDistance travelled: "<<distance<<endl;
    cout<<"\nNo. of lamp post: "<<no_lampPost<<endl;

    system("pause");
    return 0;
}
```

PROBLEM E	SEQUENCE NUMBER

Sequence number consists of series of numbers in any order. Amongst the most popular sequence numbers are Arithmetic and Geometric.

Arithmetic sequence numbers are consistently having the same difference between the consecutive number in a series. Given the following sequence number and you will realize that the difference of each consecutive number in a series is always consistently the same.

7 11 15 19 23 27 31 35 39 43

11 - 7 15 – 11 19 - 15 23 – 19 27 - 23 31 – 27 35 - 31 39 – 35 43 – 39

4 4 4 4 4 4 4 4 4

Geometric sequence numbers are consistently having the same ratio when the current number divided by the previous number. The following example shows the geometric sequence numbers consistently having the same ratio proceeding the series of numbers.

2 4 8 16 32 64 128 256 512 1024

4/2 8/4 16/8 32/16 64/32 128/64 256/128 512/256 1024/512

2 2 2 2 2 2 2 2 2

The purpose of this problem is to determine the sequence number is either Arithmetic or Geometric sequence.

Input

The first line of input contains an integer number N (1 ≤ N ≤ 5) represents the number of test cases. Following the first line are the test cases. Each line of test case contains series of integer numbers separated by comma (,). Number of item for each set of test cases is less than 10 integer numbers.

The input must be read from standard input.

PROBLEM E	SEQUENCE NUMBER (CONT')

Output

The output of the program should display as shown in the following sample of output.

Display each test case either `Arithmetic`, `Geometric` or `Invalid sequence` number.

The output must be written to standard output.

Sample Input	Sample Output

```
5
6,9,12,15,18,21,24,27
7,14,28,56,112,224,448,896
100,90,80,70,50,40,30
135,100,65,30,-5,-40
2000,1000,500,250,125
```
	<pre>
	Arithmetic Sequence
	Geometric Sequence
	Invalid sequence
	Arithmetic Sequence
	Geometric Sequence</pre>

PROBLEM E	SEQUENCE NUMBER

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

public class SequenceNumber {
public static void main(String[] args)
{
Scanner scan = new Scanner(System.in);
String lineSeparator = System.getProperty("line.separator");
scan.useDelimiter(lineSeparator);

final int totSets = 5;
String[] sequenceNumber = new String[totSets];
float [] number = new float[100];
float [] dif = new float[100];
float [] div = new float[100];
System.out.println("\n Enter "+totSets+" sets of sequence numbers : ");

for (int i=0;i<totSets;i++) {
	sequenceNumber[i] = scan.next();
	StringTokenizer token = new StringTokenizer(sequenceNumber[i],",");
	if (token.countTokens() > 10){
		System.out.println("Exceed 10 numbers ");
		sequenceNumber[i] = scan.next();
	}
}

for (int i=0;i<totSets;i++) {
	StringTokenizer token = new StringTokenizer(sequenceNumber[i],",");
	int cntToken = token.countTokens();

	for (int x=0;x<cntToken;x++)
	{	number[x]= Float.parseFloat(token.nextToken());
	}

	//arithmetic sequence
	for (int x=0;x<cntToken;x++)
	{
		dif[x] = number[x+1] - number[x];
		if (x == cntToken-1)
			dif[x] = number[x] - number[x-1];
	}
	boolean yes1 = false;
	for (int x=0;x<cntToken-1;x++)
	{
		if (dif[x]==dif[x+1])
			yes1 = true;
		else
			{yes1 = false;
			break;
```

```
                        }
        }

        if (yes1 == true)
                System.out.println("\n Arithmetic Sequence ");

        //geometric sequence
        for (int x=0;x<cntToken;x++)
        {
                div[x] = number[x+1] / number[x];
                if (x == cntToken-1)
                        div[x] = number[x] / number[x-1];
        }

        boolean yes2 = false;
        for (int x=0;x<cntToken-1;x++)
        {
                if (div[x]==div[x+1])
                        yes2 = true;
                else
                        {yes2 = false;
                        break;
                        }
        }

        if (yes2 == true)
                System.out.println("\n Geometric Sequence ");


        if ((yes1 == false) && (yes2 == false))
                System.out.println("\n Invalid sequence ");

        yes1 = false;
        yes2 = false;
}}}
```

PROBLEM F	**ARRAY ROTATION**

To rotate an array elements means to move all elements in the array from the beginning and transfer them to the end according to the number of movements required by user.

Input

The first line of the input contains 5 integers stored in an array. Following the first line is the number of movements the array should perform.

The input must be read from standard input.

Output

The output of the program should print the array's elements after the rotation.

The output must be written to standard output.

Sample Input	Sample Output

`1 2 3 4 5`
`2`
	`3 4 5 1 2`

`10 20 30 40 50`
`3`
	`40 50 10 20 30`

`100 150 200 250 300`
`4`
	`300 100 150 200 250`

PROBLEM F	ARRAY ROTATION

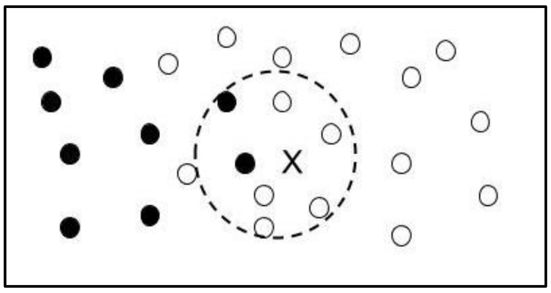
```
import java.util.*;

public class RotateString
{
 public static void main(String[] args)
 {
  Scanner in = new Scanner(System.in);
  int order;
  int arr[] = new int[5];

  for (int a = 0; a <5; a++)
  {
   arr[a] = in.nextInt();
  }

  order=in.nextInt();
  for (int i = 1; i <= order; i++) {
   for (int j = 0; j < (arr.length-1);  j++) {
    int temp = arr[j];
    arr[j] = arr[j +1];
    arr[j+ 1] = temp;
   }
  }
  for (int x = 0; x <arr.length; x++)
   System.out.print(arr[x] + " ");
 }
}
```

PROBLEM G	**k-NN**

In Artificial Intelligence, there is a subset field called machine learning which provides the computer the ability to automatically learn and improve from experience without being explicitly programmed.

One of such simple and very popular technique is k-NN (K-Nearest Neighbours).

k-NN works by searching the k nearest points (neighbours) to the desired point. By getting the nearest points, k-NN will try to make an assumption based on the nearest points' classes and use that to predict the class of the desired point. In A.I. this is also called classifying problem as we want to predict the class of the desired point.



k = 7

For example, above is set of points with two classes (A = black circle, B = transparent circle). X is a point that we desired to predict. k is number of nearest points that we want to find (which in this case is 7). By searching the distances between X and all points, we then sort out the result. Based on the nearest seven points to the X, we found out that there are 2 and in class A and 5 are in the class B. By getting this information, k-NN will choose the highest frequency among all the classes and uses that determine class of X (which in case X = class B).

Input

First line of input has P (5 ≤ P ≤ 20), which represent number of points in the cartesian plane. For each P lines, there will contain three input X, Y, C where the first two input represent the coordinate of the point and C represent its class.

X (0 ≤ X ≤ 10), Y (0 ≤ Y ≤ 10), C (1 ≤ C ≤ 2).

Next line will contain T (1 ≤ P ≤ 5), where it represents the number of test case(s). For each next T lines, there will be three value I, J and K, where I and J represent location of point where we want to predict, and K represent how much nearest nodes we want to compare.

I (0 ≤ I ≤ 10), J (0 ≤ J ≤ 10), K (1 ≤ K ≤ P) and K is odd number.

PROBLEM G	k-NN (cont')

Output

Print the the case number ‘Case #n:’; n is test case number followed by class predicted by your k-NN algorithm. Please note that there will be only two classes, 1 and 2.

Sample Input	Sample Output

`5`
`4 2 1`
`7 2 2`
`3 6 1`
`7 4 2`
`2 3 1`
`3`
`3 4 1`
`5 4 3`
`4 4 5`
	`Case #1: 1`
	`Case #2: 2`
	`Case #3: 1`

Hint:

$$\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$$

PROBLEM G	k-NN

```
#include <iostream>
#include <algorithm>
#include <cmath>

using namespace std;

struct Point
{
    int x, y;
    int pclass;
};

struct PDistance
{
    Point point;
    double distance;
};

double distance(int xd, int yd) {
    return sqrt(pow(xd,2) + pow(yd,2));
}

bool cmp(const PDistance &a, const PDistance &b)
{
    return a.distance < b.distance;
}

int main()
{
    int P;
    cin >> P;
    Point points[P];
    for (int i=0; i<P; i++) {
        cin >> points[i].x >> points[i].y >> points[i].pclass;
    }

    int T;
    cin >> T;
    for (int i=0; i<T; i++) {

        int I, J, K;
        PDistance pdistance[P];
        cin >> I >> J >> K;

        for (int j=0; j<P; j++) {
            pdistance[j].point = points[j];
            pdistance[j].distance = distance(I-points[j].x, J-
points[j].y);
        }

        sort(pdistance, pdistance+P, cmp);

        int count[3] = {0};
        for (int j=0; j<K; j++) {
            count[pdistance[j].point.pclass]++;
        }

        cout << "Case #" << (i+1) << ": " << (count[1] > count[2] ? 1 :
2) << endl;
    }

    return 0;
}
```

PROBLEM H	**ODD & EVEN**

Write a program that is able to list out all the even and odd numbers in a number ranging from 100,000 to 999,999.

Input:

An integer number from 100,000 to 999,000.

Output:

2 lists of numbers; 1 list of even numbers and another list of odd numbers.

Sample input	Sample output

`678543`	`6 8 4`
	`7 5 3`

`999888`	`8 8 8`
	`9 9 9`

PROBLEM H	ODD & EVEN

```
#include<iostream>
using namespace std;

int main()
{
    //input a number between 100,000 to 999,999
    char number[6];
    //cout<<"Enter a number between 100,000 to 999,999: ";
    cin>>number;

    int no;
    char even[6];
    char odd[6];
    int indexEven = 0;
    int indexOdd = 0;
    //display all the even number & the odd numbers
    for(int index = 0; index < 6; index++)
    {
            no = (int) number[index]- 48;
            //cout<<no<<endl;
            if(no % 2 == 0) //even
            {
                  even[indexEven] = number[index];
                  indexEven++;
            }
            else
            {
                  odd[indexOdd] = number[index];
                  indexOdd++;
            }

    }

    //cout<<"The list of even number is ";
    for(int counter = 0; counter < indexEven; counter++)
            cout<<even[counter]<<" ";

    cout<<"\n";
    for(int counter = 0; counter < indexOdd; counter++)
            cout<<odd[counter]<<" ";

    system("pause");
    return 0;
}
```

PROBLEM I Algebra

Write a program to solve a n x n linear system of simultaneous equations. For example:

$$x + 2y + 3z = 14$$

$$2x + 3y + 4z = 20$$

$$3x + 4y + 4z = 23$$

In short, the above equations can be represented as augmented matrix. An augmented matrix for a system of equations is a matrix of numbers in which each row represents the constants from one equation (both the coefficients and the constant on the other side of the equal sign) and each column represents all the coefficients for a single variable. The first row consists of all the constants from the first equation with the coefficient of the *x* in the first column, the coefficient of the *y* in the second column, the coefficient of the *z* in the third column and the constant in the final column.

$$\left(\begin{array}{ccc|c} 1 & 2 & 3 & 14 \\ 2 & 3 & 4 & 20 \\ 3 & 4 & 4 & 23 \end{array}\right)$$

Solve above equation will get x = 1, y = 2 and z = 3. Using this value, you can solve the above equations.

Input

The first line of the input contains an integer T *(T ≤ 100),* number of test cases. Each of the following T lines will input the size of matrix n (*n ≤ 10),* the following n lines will contain n+1 value which in the form of augmented matrix.

Output

For each test case, the output contains a line in the format Case #x: M, where x is the case number (starting from 1) and M is the variable values in one decimal point separated by space.

PROBLEM I	Algebra (cont’)

Sample input

```
3
2
4 3 7
1 1 -1
3
1 2 3 14
2 3 4 20
3 4 4 23
3
1 1 1 25
5 3 2 0
0 1 -1 6
```

Sample output

```
Case #1: 10 -11
Case #2: 1 2 3
Case #3: -26.2 28.6 22.6
```

PROBLEM I	Algebra (cont’)

```
import java.util.Scanner;
import java.util.Arrays;
import java.lang.Math;
import java.text.DecimalFormat;


public class algebra {
  public static void main(String [] args) {
    Scanner scn = new Scanner(System.in);
    String ls = System.getProperty("line.separator");
    scn.useDelimiter(ls);

    // get number of test case
    int n = scn.nextInt();

    for (int i =0; i < n; i++) {
      // get the matrix dimension m x m
      int m = scn.nextInt();

      // create matrix size m x m with zero values
      // plus 1 column for output
      double [][] matrix = new double[m][m+1];

      // get the input value for matrix
      for (int j = 0; j < m; j++) {
        String s = scn.next();
        String [] s_array = s.split(" ");

        for (int k = 0; k < s_array.length; k++) {
          matrix[j][k] =  Double.parseDouble(s_array[k]);
        }
      }

      // there are 3 steps to solve linear system,
      // 1. swap row
      // 2. add row with other row
      // 3. multiply with constant

      for (int j = 0; j < m; j++) {
        // search maximum value in current column

        double maxVal = Math.abs(matrix[j][j]);
        int maxRow = j;
        for (int k = j+1; k < m; k++) {
          double val = Math.abs(matrix[k][j]);

          if (maxVal < val) {

            maxVal = val;
            maxRow = k;
          }
        }

        // based on maximum row, we swap it with current row
        for (int k = j; k < m+1; k++) {
          double temp = matrix[maxRow][k];
          matrix[maxRow][k] = matrix[j][k];
          matrix[j][k] = temp;
        }

        // convert matrix into row echelon form

        for (int k = j+1; k < m; k++) {
          double constant = -matrix[k][j] / matrix[j][j];
          for (int l = j; l < m+1; l++) {
            if (j == l) {
              matrix[k][l] = 0;
            }
            else {
              matrix[k][l] += constant * matrix[j][l];
            }
          }
        }
      }

      double [] coefficient = new double[m];

      for (int j = m-1; j>=0; j--) {
        coefficient[j] = matrix[j][m]/matrix[j][j];
        for (int k = j-1; k>=0; k--) {
          matrix[k][m] -= matrix[k][j] * coefficient[j];
        }
      }

      DecimalFormat df = new DecimalFormat("#.##");
      System.out.print("Case #"+(i+1)+": ");
      for (int j = 0; j < coefficient.length; j++) {
        System.out.print(df.format(coefficient[j])+' ');
      }
      System.out.println();
    }
  }
}
```

PROBLEM J	QUOTIENT AND REMAINDER

When we divide two integer numbers, we will get a quotient and a remainder. You are asked to write a program to find a quotient and a remainder, when a bigger number is being divided by the smaller number (do not assume that the numbers are entered in any order). For example, when numbers 4 and 2 are entered, the answer is Q 2 R 0 or, when 2 and 7 are entered, it will give the answer is Q 3 R 1 (where, Q represents a quotient and R represents a remainder.

Input

The first line of the input contains an integer N (1 ≤ N ≤ 5), the number of test cases. Following the first line are the test cases. Each line of a test case contains two non-negative integer numbers.

The input must be read from standard input.

Output

The output of the program should display the answer in the form Q <quotient> R<remainder>.

The output must be written to standard output.

Sample Input	Sample Output

```
3
5   8
4   2
8   10
```
	<pre>
	Q 1 R 3
	Q 2 R 0
	Q 1 R 2</pre>

PROBLEM J	QUOTIENT AND REMAINDER

```
using namespace std;

int main()
{
      int x, no1, no2, q, r, noCase;

      cin>>noCase;

      for(x = 0; x<noCase; x++){

      cin>>no1>>no2;

      if(no1 > no2)
      {
            q = no1/no2;
            r = no1%no2;
      }

      else
         if(no1 < no2)
         {
                  q = no2/no1;
                  r = no2%no1;
         }

      cout<<"Q "<<q<< " R "<<r<<endl;
}
}
```