# Assignment No. - 1

**//Problem Statement :** Problems in Computer Science are often classified as belonging to a certain class of problems (e.g., NP, Unsolvable, Recursive). In this problem you will be analyzing a property of an algorithm whose classification is not known for all possible inputs.
Consider the following algorithm:

```
1. input n
2. 2. print n
3. 3. if n = 1 then STOP
4. 4. if n is odd then n ←− 3n + 1
5. 5. else n ←− n/2
6. 6. GOTO 2
```

Given the input 22, the following sequence of numbers will be printed
22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
It is conjectured that the algorithm above will terminate (when a 1 is printed) for any integral input value. Despite the simplicity of the algorithm, it is unknown whether this conjecture is true. It has been verified, however, for all integers n such that 0 < n < 1, 000, 000 (and, in fact, for many more numbers than this.) Given an input n, it is possible to determine the number of numbers printed before and including the 1 is printed. For a given n this is called the cycle-length of n. In the example above, the cycle length of 22 is 16. For any two numbers i and j you are to determine the maximum cycle length over all numbers between and including both i and j.

**Input**

The input will consist of a series of pairs of integers i and j, one pair of integers per line. All integers will be less than 10,000 and greater than 0. You should process all pairs of integers and for each pair determine the maximum cycle length over all integers between and including i and j. You can assume that no operation overflows a 32-bit integer.

**Output**

For each pair of input integers i and j you should output i, j, and the maximum cycle length for integers between and including i and j. These three numbers should be separated by at least one space with all three numbers on one line and with one line of output for each line of input. The integers i and j must appear in the output in the same order in which they appeared in the input and should be followed by the maximum cycle length (on the same line).

## Assignment No. - 1

**//Developed By : Yash Mete**

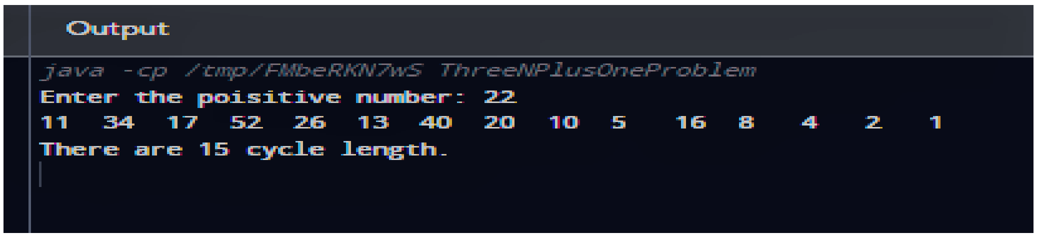
**//AIM** : To Solve the 3N+1 problem

**//Date** : 21/03/2023

**//Usage** : Given a positive integer, N, define the 3N+1 sequence starting from N as follows: If N is an even number, then divide N by two. If N is an odd number, then multiply N by 3 and add 1. Continue to generate numbers in this way until N becomes equal to 1.

**//Program :**
```
import java.util.Scanner;
public class ThreeNPlusOneProblem
{
public static void main(String args[])
{
int N;
int count;
Scanner sc=new Scanner(System.in);
System.out.print("Enter the starting point for the sequence of
the numbers: ");
N=sc.nextInt();
while (N <= 0)
{
System.out.println("Please re-enter the number beacuse The
starting point must be positive.: ");
N=sc.nextInt();
}
count = 0;
while (N != 1)
{
if (N % 2 == 0)
N = N / 2;
else
N = 3 * N + 1;
System.out.print(N + "\t");
count = count + 1;
}
System.out.println();
System.out.println("There are "+count+" terms in the sequence
of the numbers.");
}
}
```

**//Output :**



```
java -cp /tmp/FMbeRKN7wS ThreeNPlusOneProblem
Enter the poisitive number: 22
11   34   17   52   26   13   40   20   10   5    16   8    4    2    1
There are 15 cycle length.
```

## Assignment No. - 2

**//Problem Statement :** A number of students are members of a club that travels annually to exotic locations. Their destinations in the past have included Indianapolis, Phoenix, Nashville, Philadelphia, San Jose, and Atlanta. This spring they are planning a trip to Eindhoven. The group agrees in advance to share expenses equally, but it is not practical to have them share every expense as it occurs. So individuals in the group pay for particular things, like meals, hotels, taxi rides, plane tickets, etc. After the trip, each student's expenses are tallied and money is exchanged so that the net cost to each is the same, to within one cent. In the past, this money exchange has been tedious and time consuming. Your job is to compute, from a list of expenses, the minimum amount of money that must change hands in order to equalize (within a cent) all the students' costs.

**Input**

Standard input will contain the information for several trips. The information for each trip consists of a line containing a positive integer, n, the number of students on the trip, followed by n lines of input, each containing the amount, in dollars and cents, spent by a student. There are no more than 1000 students and no student spent more than $10,000.00. A single line containing 0 follows the information for the last trip.

**Output**

For each trip, output a line stating the total amount of money, in dollars and cents, that must be exchanged to equalize the students' costs.

**Assignment No. - 2**

**//Developed By : Yash Mete**

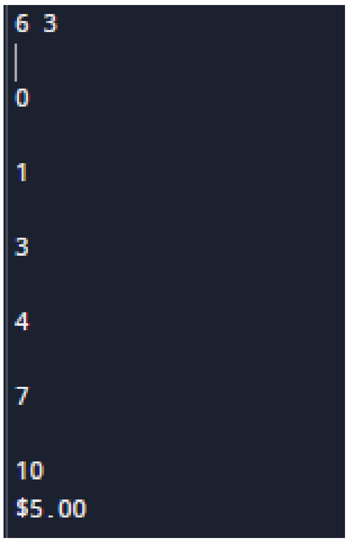
**//AIM :** To Solve the The Trip problem

**//Date :** 21/03/2023

**//Usage :** Given the trip problem gives us  The least number of stops, and the number of ways to plan the trip which uses the least number of stops

**//Program :**
```
import java.util.*;
public class The Trip {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int[] expenses = new int[n];
        double total = 0.0;
        for (int i = 0; i < n; i++) {
            expenses[i] = sc.nextInt();
            total += expenses[i];
        }
        double average = total / n;
        double difference = 0.0;
        for (int i = 0; i < n; i++) {
            difference += Math.max(0, expenses[i] - average);
        }
        System.out.printf("$%.2f\n", difference);
    }
}
```

**//Output :**



```
6 3

0

1

3

4

7

10
$5.00
```

## Assignment No. - 3

**//Problem Statement :** A friend of you has just bought a new computer. Until now, the most powerful computer he ever used has been a pocket calculator. Now, looking at his new computer, he is a bit disappointed, because he liked the LC-display of his calculator so much. So you decide to write a program that displays numbers in an LC-display-like style on his computer.

**Input**

The input file contains several lines, one for each number to be displayed. Each line contains two integers s, n ($1 \leq s \leq 10$, $0 \leq n \leq 99\ 999\ 999$), where n is the number to be displayed and s is the size in which it shall be displayed. The input file will be terminated by a line containing two zeros. This line should not be processed.

**Output**

Output the numbers given in the input file in an LC-display-style using s '-' signs for the horizontal segments and s '|' signs for the vertical ones. Each digit occupies exactly s + 2 columns and 2s + 3 rows. (Be sure to fill all the white space occupied by the digits with blanks, also for the last digit.) There has to be exactly one column of blanks between two digits. Output a blank line after each number. (You will find a sample of each digit in the sample output.)

# Assignment No. - 3

**//Developed By : Yash Mete**

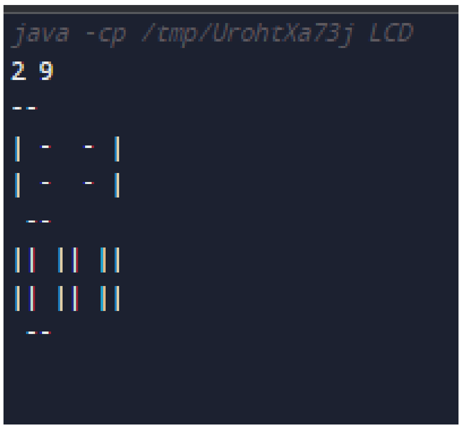
**//AIM :** To Solve the LCD-Display problem

**//Date :** 28/03/2023

**//Usage :** The LCD display or video issue can occur due to outdated drivers such as BIOS, video card (GPU), chipset, and monitor driver, video, or graphic settings in the operating system, faulty video cable, outdated operating system updates.

**//Program :**
```
import java.util.*;
public class LCD {
    private static final String[][] DIGITS = {
        {" - ", "| |", "   ", "| |", " - "}, // 0
        {"   ", "  |", "   ", "  |", "   "}, // 1
        {" - ", "  |", " - ", "|  ", " - "}, // 2
        {" - ", "  |", " - ", "  |", " - "}, // 3
        {"   ", "| |", " - ", "  |", "   "}, // 4
        {" - ", "|  ", " - ", "  |", " - "}, // 5
        {" - ", "|  ", " - ", "| |", " - "}, // 6
        {" - ", "  |", "   ", "  |", "   "}, // 7
        {" - ", "| |", " - ", "| |", " - "}, // 8
        {" - ", "| |", " - ", "  |", " - "}  // 9
    };
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        while (true) {
            int s = sc.nextInt();
            String n = sc.next();

            if (s == 0 && n.equals("0")) {
                break;
            }
            int rows = 2 * s + 3;
            int cols = s + 2;
            String[][] display = new String[rows][cols *
n.length()];
            for (int i = 0; i < n.length(); i++) {
                int digit =
Character.getNumericValue(n.charAt(i));
                String[] digitDisplay = DIGITS[digit];
                for (int j = 0; j < rows; j++) {
                    for (int k = 0; k < cols; k++) {
                        String symbol = digitDisplay[j / (s +
1)];
                        if (j % (s + 1) == 0) {
                            symbol = (k == 0 || k == cols -
1) ? " " : "-";
                        } else if (k == 0) {
                            symbol = "|";
                        } else if (k == cols - 1) {
                            symbol = "|";
                        }
                        display[j][i * cols + k] = symbol;
                    }
                }
            }
            for (String[] row : display) {
                for (String symbol : row) {
                    System.out.print(symbol);
                }
                System.out.println();
            }
            System.out.println();
        }
    }
}
```

**//Output :**



```
java -cp /tmp/UrohtXa73j LCD
2 9
--
| -  - |
| -  - |
 --
|| || ||
|| || ||
 --
```

## Assignment No. - 4

**//Problem Statement :** The simple graphical editor deals with a rectangular table M × N (1 ≤ M, N ≤ 250). Each pixel of the table has its colour. The picture is formed from this square pixels. The problem is to write a program, which simulates an interactive work of the graphical editor.

**Input**

Input consists of the editor commands, one per line. Each command is represented by one Latin capital placed in the very beginning of the line. If the command supposes parameters, all the parameters will be given in the same line separated by space. As the parameters there may be: the coordinates of the pixel - two integers, the first one is the column number and belongs to 1 . . . M, the second one is the row number and belongs to 1 . . . N, the origin is in the upper left corner of the table; the colour - the Latin capital; file name - in MSDOS 8.3 format. The editor deals with the following commands:
I M N Creates a new table M × N. All the pixels are colored in white (O).
C Clears the table. The size remains the same. All the pixels became white (O).
L X Y C Colors the pixel with coordinates (X, Y ) in colour C.
V X Y1 Y2 C Draws the vertical segment in the column X between the rows Y1 and Y2 inclusive in colour C.
H X1 X2 Y C Draws the horizontal segment in the row Y between the columns X1 and X2 inclusive in colour C.
K X1 Y1 X2 Y2 C Draws the filled rectangle in colour C. (X1, Y1) is the upper left corner, (X2, Y2) is the lower right corner of the rectangle.
F X Y C Fills the region with the colour C. The region R to be filled is defined as follows. The pixel (X, Y ) belongs to this region. The other pixel belongs to the region R if and only if it has the same colour as pixel (X, Y ) and a common side with any pixel which belongs to this region.
S N ame Writes the picture in the file N ame.
X Terminates the session.

**Output**

Every time the command 'S NAME' meets, you should output the file name NAME and the current table, row by row. Each row is represented by a pixels' colours series, see the output sample.
Errors: If as a command there will be a character different from I, C, L, V, H, K, F, S, X, the editor should ignore the whole line and pass to the next command. In case of other errors the program behaviour is unpredictable.

## Assignment No. - 4

**//Developed By : Yash Mete**

**//AIM :** To Solve the Graphical Editor problem

**//Date :** 28/03/2023

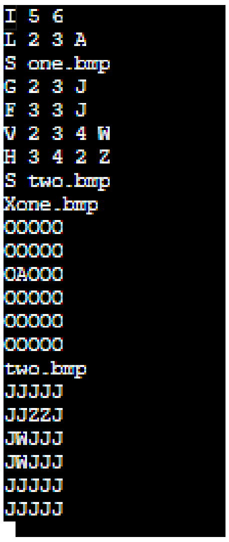
**//Usage :** Graphic editors are computer software that allows users to manipulate and edit graphical images for use in various areas. The software either comes with a computer like the graphic editor mac or can be purchased from a developer.

**//Program :**

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

class Main
{
	static char mat[][]=new char[1][1];
	public static void main(String args[])throws IOException
	{
		BufferedReader lee = new BufferedReader(new
InputStreamReader(System.in));
		String s;
		StringTokenizer str;
		while(!(s=lee.readLine()).equals("X"))
		{
			char K;
			char C;
			str=new StringTokenizer(s);
			C=str.nextToken().charAt(0);
			if(C=='I')
			{
				int p=Integer.parseInt(str.nextToken());
				mat=new
char[Integer.parseInt(str.nextToken())][p];
				Fill();
			}
			else if(C=='C')Fill();
			else if(C=='L')
			{
				int X=Integer.parseInt(str.nextToken())-1;
				int Y=Integer.parseInt(str.nextToken())-1;
				mat[Y][X]=str.nextToken().charAt(0);
			}
			else if(C=='V')
			{
				int col=Integer.parseInt(str.nextToken())-1;
				int A=Integer.parseInt(str.nextToken())-1;
				int B=Integer.parseInt(str.nextToken())-1;
				if(A>B)
				{
					int aux=A;
```

```
                            A=B;
                            B=aux;
                    }
                    K=str.nextToken().charAt(0);
                    for(int i=A;i<=B;i++)mat[i][col]=K;
            }
            else if(C=='H')
            {
                    int A=Integer.parseInt(str.nextToken())-1;
                    int B=Integer.parseInt(str.nextToken())-1;
                    int row=Integer.parseInt(str.nextToken())-1;
                    if(A>B)
                    {
                            int aux=A;
                            A=B;
                            B=aux;
                    }
                    K=str.nextToken().charAt(0);
                    for(int i=A;i<=B;i++)
                            mat[row][i]=K;
            }
            else if(C=='K')
            {
                    int A=Integer.parseInt(str.nextToken())-1;
                    int B=Integer.parseInt(str.nextToken())-1;
                    int D=Integer.parseInt(str.nextToken())-1;
                    int E=Integer.parseInt(str.nextToken())-1;
                    K=str.nextToken().charAt(0);
                    for(int i=B;i<=E;i++)
                            for(int j=A;j<=D;j++)
                                    mat[i][j]=K;
            }
            else if(C=='F')
            {
                    int X=Integer.parseInt(str.nextToken())-1;
                    int Y=Integer.parseInt(str.nextToken())-1;
                    char colorear=mat[Y][X];
                    K=str.nextToken().charAt(0);
                    if(K==colorear)continue;
                    Floooooood(X,Y,K,colorear);

            }
            else if(C=='S')
            {
                    System.out.println(str.nextToken());
                    for(int i=0;i<mat.length;i++)
                    {
                            for(int j=0;j<mat[0].length;j++)
                                    System.out.print(mat[i][j]);
                            System.out.println();
                    }
            }
    }
}
public static void Floooooood(int X,int Y, char K, char colorear)
{
		mat[Y][X]=K;
		if(X+1<mat[0].length&&mat[Y][X+1]==colorear)
		{
			Floooooood(X+1,Y,K,colorear);
		}
		if(Y+1<mat.length&&mat[Y+1][X]==colorear)
		{
			Floooooood(X,Y+1,K,colorear);
		}
		if(X-1>=0&&mat[Y][X-1]==colorear)
		{
			Floooooood(X-1,Y,K,colorear);
		}
		if(Y-1>=0&&mat[Y-1][X]==colorear)
		{
			Floooooood(X,Y-1,K,colorear);
		}

	}


	public static void Fill()
	{
		for(int i=0;i<mat.length;i++)
			for(int j=0;j<mat[0].length;j++)
				mat[i][j]='O';
	}
}
```

//Output :


```
I 5 6
L 2 3 A
S one.bmp
G 2 3 J
F 3 3 J
V 2 3 4 W
H 3 4 2 Z
S two.bmp
Xone.bmp
OOOOO
OOOOO
OAOOO
OOOOO
OOOOO
OOOOO
two.bmp
JJJJJ
JJZZJ
JWJJJ
JWJJJ
JJJJJ
JJJJJ
```

## Assignment No. - 5

**//Problem Statement :** Australian ballots require that the voter rank the candidates in order of choice. Initially only the first choices are counted and if one candidate receives more than 50% of the vote, that candidate is elected. If no candidate receives more than 50%, all candidates tied for the lowest number of votes are eliminated. Ballots ranking these candidates first are recounted in favour of their highest ranked candidate who has not been eliminated. This process continues [that is, the lowest candidate is eliminated and each ballot is counted in favour of its ranked non-eliminated candidate] until one candidate receives more than 50% of the vote or until all candidates are tied.

**Input**

The input begins with a single positive integer on a line by itself indicating the number of the cases following, each of them as described below. This line is followed by a blank line, and there is also a blank line between two consecutive inputs. The first line of input is an integer n ≤ 20 indicating the number of candidates. The next n lines consist of the names of the candidates in order. Names may be up to 80 characters in length and may contain any printable characters. Up to 1000 lines follow; each contains the contents of a ballot. That is, each contains the numbers from 1 to n in some order. The first number indicates the candidate of first choice; the second number indicates candidate of second choice, and so on.

**Output**

For each test case, the output must follow the description below. The outputs of two consecutive cases will be separated by a blank line. The Output consists of either a single line containing the name of the winner or several lines containing the names of the candidates who tied.

**Assignment No. - 5**

**//Developed By :** Yash Mete

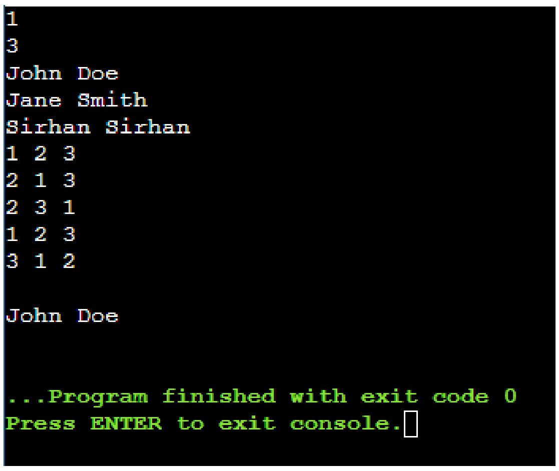
**//AIM :** To Solve the Austrlian Voting problem

**//Date :** 28/03/2023

**//Usage :** Voting is compulsory at federal elections, by-elections and referendums for those on the electoral roll, as well as for State and Territory elections. Australia enforces compulsory voting. People in this situation are asked to explain their failure to vote.so this Austrlian voting helps us.

**//Program :**
```java
import java.util.*;
public class AustralianVoting {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int t = sc.nextInt();
        sc.nextLine();
        for (int i = 0; i < t; i++) {
            int n = sc.nextInt();
            sc.nextLine();
            String[] candidates = new String[n];
            for (int j = 0; j < n; j++) {
                candidates[j] = sc.nextLine();
            }
            int[][] votes = new int[n][n];
            while (true) {
                String line = sc.nextLine();
                if (line.equals("")) {
                    break;
                }
                String[] parts = line.split(" ");
                for (int j = 0; j < n; j++) {
                    votes[Integer.parseInt(parts[j]) - 1][j]++;
                }
            }
            int[] eliminated = new int[n];
            int remaining = n;
            while (true) {
                int[] firstPlaceVotes = new int[n];
                for (int j = 0; j < n; j++) {
                    if (eliminated[j] == 0) {
                        firstPlaceVotes[j] = votes[j][0];
                    }
                }
                boolean majority = false;
                for (int j = 0; j < n; j++) {
                    if  (eliminated[j]  ==  0  &&  firstPlaceVotes[j]  >
remaining / 2) {
                        System.out.println(candidates[j]);
                        majority = true;
                        break;
                    }
                }
                if (majority) {
                    break;
                }
                int minVotes = Integer.MAX_VALUE;
                for (int j = 0; j < n; j++) {
                    if  (eliminated[j]  ==  0  &&  firstPlaceVotes[j]  <
minVotes) {
                        minVotes = firstPlaceVotes[j];
                    }
                }
                if (remaining == 1) {
                    for (int j = 0; j < n; j++) {
                        if (eliminated[j] == 0) {
                            System.out.println(candidates[j]);
                            break;
                        }
                    }
                    break;
                }
                for (int j = 0; j < n; j++) {
                    if  (eliminated[j]  ==  0  &&  firstPlaceVotes[j]  ==
minVotes) {
                        eliminated[j] = 1;
                        remaining--;
                    }
                }
                for (int j = 0; j < n; j++) {
                    for (int k = 0; k < n; k++) {
                        if (eliminated[k] == 1) {
                            continue;
                        }

                        int voteIndex = 0;
                        while      (voteIndex      <      n      &&
eliminated[votes[k][voteIndex] - 1] == 1) {
                            voteIndex++;
                        }

                        if (voteIndex < n) {
                            votes[k][voteIndex]++;
                        }
                    }
                }
            }

            if (i != t - 1) {
                System.out.println();}}}}
```

```
//Output :
```



```
1
3
John Doe
Jane Smith
Sirhan Sirhan
1 2 3
2 1 3
2 3 1
1 2 3
3 1 2

John Doe


...Program finished with exit code 0
Press ENTER to exit console.
```

## Assignment No. - 6

**//Problem Statement -** A sequence of n > 0 integers is called a jolly jumper if the absolute values of the difference between successive elements take on all the values 1 through n – 1. For instance,
 1 4 2 3
 is a jolly jumper, because the absolutes differences are 3, 2, and 1 respectively. The definition implies that any sequence of a single integer is a jolly jumper. You are to write a program to determine whether or not each of a number of sequences is a jolly jumper.

**Input**

Each line of input contains an integer n ≤ 3000 followed by n integers representing the sequence.

**Output**

For each line of input, generate a line of output saying 'Jolly' or 'Not jolly'.

**Assignment No. - 6**

**//Developed By :** Yash Mete

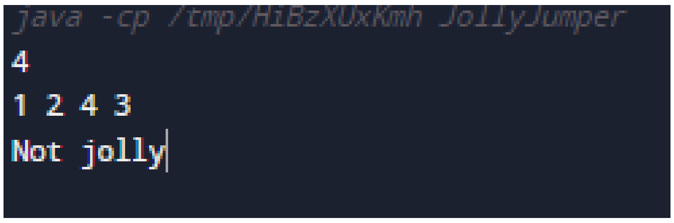
**//AIM :** To Solve the Jolly Jumper problem

**//Date :** 03/04/2023

**//Usage :** The device holds a baby upright in a seat or harness so only their toes are touching the floor. The baby can then bounce up and down.

**//Program :**
```
import java.util.*;
public class JollyJumper {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int[] sequence = new int[n];
        boolean[] diff = new boolean[n - 1];
        for (int i = 0; i < n; i++) {
            sequence[i] = sc.nextInt();
        }
        for (int i = 1; i < n; i++) {
            int   absoluteDiff   =   Math.abs(sequence[i]   -
sequence[i-1]);
            if (absoluteDiff < 1 || absoluteDiff > n - 1 ||
diff[absoluteDiff-1]) {
                System.out.println("Not jolly");
                return;
            }
            diff[absoluteDiff-1] = true;
        }
        System.out.println("Jolly");
    }
}
```

// Output :



```
java -cp /tmp/HiBzXUxKmh JollyJumper
4
1 2 4 3
Not jolly
```

**Assignment No. - 7**

**//Problem Statement :** A social research organization has determined a simple set of parameters to simulate the behavior of the political parties of our country. One of the parameters is a positive integer h (called the hartal parameter) that denotes the average number of days between two successive hartals (strikes) called by the corresponding party. Though the parameter is far too simple to be flawless, it can still be used to forecast the damages caused by hartals. The following example will give you a clear idea: Consider three political parties. Assume h1 = 3, h2 = 4 and h3 = 8 where hi is the hartal parameter for party i (i = 1, 2, 3). Now, we will simulate the behavior of these three parties for N = 14 days. One must always start the simulation on a Sunday and assume that there will be no hartals on weekly holidays (on Fridays and Saturdays).

```
      1  2  3  4  5  6  7  8  9  10 11 12 13 14
 Days Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa
Party 1 x x x x
 Party 2 x x x
Party 3 x
Hartals 1 2 3 4 5
```

The simulation above shows that there will be exactly 5 hartals (on days 3, 4, 8, 9 and 12) in 14 days. There will be no hartal on day 6 since it is a Friday. Hence we lose 5 working days in 2 weeks. In this problem, given the hartal parameters for several political parties and the value of N, your job is to determine the number of working days we lose in those N days.

**Input**

The first line of the input consists of a single integer T giving the number of test cases to follow. The first line of each test case contains an integer N (7 ≤ N ≤ 3650) giving the number of days over which the simulation must be run. The next line contains another integer P (1 ≤ P ≤ 100) representing the number of political parties in this case. The i th of the next P lines contains a positive integer hi (which will never be a multiple of 7) giving the hartal parameter for party i (1 ≤ i ≤ P).

**Output**

For each test case in the input output the number of working days we lose. Each output must be on a separate line.

## Assignment No. - 7

**//Developed By :** Yash Mete

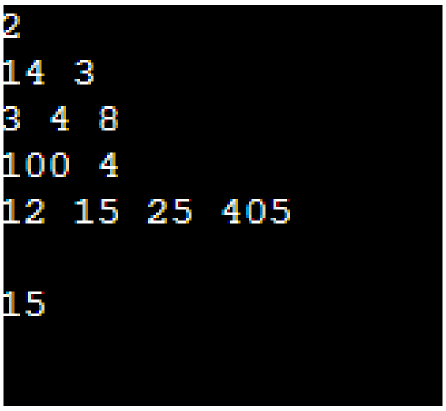
**//AIM :** To Solve the Hartals problem

**//Date :** 03/04/2023

**//Usage :** It is a mode of appealing to the sympathies of a government to reverse an unpopular or unacceptable decision. A hartal is often used for political reasons, for example by an opposition party protesting against a governmental policy or action.

**//Program :**
```java
import java.util.*;
public class Hartals {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int t = sc.nextInt();
        for (int i = 0; i < t; i++) {
            int n = sc.nextInt();
            int p = sc.nextInt();
            int[] hartals = new int[p];
            int count = 0;
            for (int j = 0; j < p; j++) {
                hartals[j] = sc.nextInt();
            }
            for (int j = 1; j <= n; j++) {
                if (j % 7 != 0 && (j + 1) % 7 != 0) {
                    boolean found = false;
                    for (int k = 0; k < p; k++) {
                        if (j % hartals[k] == 0) {
                            found = true;
                            break;
                        }
                    }
                    if (found) {
                        count++;
                    }
                }
            }
            System.out.println(count);
        }
    }
}
```

**//Output :**



```
2
14 3
3 4 8
100 4
12 15 25 405

15
```

## Assignment No. - 8

**//Problem Statement :** A common but insecure method of encrypting text is to permute the letters of the alphabet. That is, in the text, each letter of the alphabet is consistently replaced by some other letter. So as to ensure that the encryption is reversible, no two letters are replaced by the same letter. Your task is to decrypt several encoded lines of text, assuming that each line uses a different set of replacements, and that all words in the decrypted text are from a dictionary of known words.

**Input**

The input consists of a line containing an integer n, followed by n lower case words, one per line, in alphabetical order. These n words comprise the dictionary of words which may appear in the decrypted text. Following the dictionary are several lines of input. Each line is encrypted as described above. There are no more than 1000 words in the dictionary. No word exceeds 16 letters. The encrypted lines contain only lower case letters and spaces and do not exceed 80 characters in length.

**Output**

Decrypt each line and print it to standard output. If there is more than one solution, any will do. If there is no solution, replace every letter of the alphabet by an asterisk

## Assignment No. - 8

**//Developed By :** Yash Mete

**//AIM :** To Solve the Crypt Kicker problem

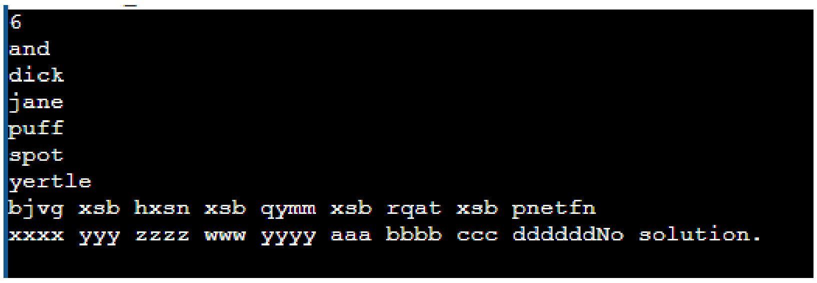
**//Date :** 03/03/2023

**//Usage :** A common but insecure method of encrypting text is to permute the letters of the alphabet. That is, in the text, each letter of the alphabet is consistently replaced by some other letter. So as to ensure that the encryption is reversible, no two letters are replaced by the same letter. Your task is to decrypt several encoded lines of text, assuming that each line uses a different set of replacements, and that all words in the decrypted text are from a dictionary of known words.

**//Program :**

```java
import java.util.*;
public class CryptKicker {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        sc.nextLine();
        String[] dict = new String[n];
        for (int i = 0; i < n; i++) {
            dict[i] = sc.nextLine();
        }
        String encrypted = sc.nextLine();
        String[] words = encrypted.split("\\s+");
        Map<Character, Character> key = new HashMap<>();
        StringBuilder sb = new StringBuilder();
        boolean valid = true;
        for (String word : words) {
            if (word.length() == 1) {
                sb.append(word).append(" ");
                continue;
            }
            String decrypted = "";
            for (char c : word.toCharArray()) {
                if (key.containsKey(c)) {
                    decrypted += key.get(c);
                } else {
                    decrypted += "_";
                }
            }
            if
(Arrays.stream(dict).anyMatch(decrypted::equals)) {
                sb.append(decrypted).append(" ");
            } else {
                valid = false;
                break;
            }
        }
        if (valid) {
            System.out.println(sb.toString().trim());
        } else {
            System.out.println("No solution.");
        }
    }
}
```

//Output :


```
6
and
dick
jane
puff
spot
yertle
bjvg xsb hxsn xsb qymm xsb rqat xsb pnetfn
xxxx yyy zzzz www yyyy aaa bbbb ccc ddddddNo solution.
```

## Assignment No. - 9

**//Problem statement :** A standard playing card deck contains 52 cards, 13 values in each of four suits. The values are named 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen, King, Ace. The suits are named Clubs, Diamonds, Hearts, Spades. A particular card in the deck can be uniquely identified by its value and suit, typically denoted < value > of < suit >. For example, "9 of Hearts" or "King of Spades". Traditionally a new deck is ordered first alphabetically by suit, then by value in the order given above. The Big City has many Casinos. In one such casino the dealer is a bit crooked. She has perfected several shuffles; each shuffle rearranges the cards in exactly the same way whenever it is used. A very simple example is the "bottom card" shuffle which removes the bottom card and places it at the top. By using various combinations of these known shuffles, the crooked dealer can arrange to stack the cards in just about any particular order. You have been retained by the security manager to track this dealer. You are given a list of all the shuffles performed by the dealer, along with visual cues that allow you to determine which shuffle she uses at any particular time. Your job is to predict the order of the cards after a sequence of shuffles.

**Input**

The input begins with a single positive integer on a line by itself indicating the number of the cases following, each of them as described below. This line is followed by a blank line, and there is also a blank line between two consecutive inputs. Input consists of an integer n ≤ 100, the number of shuffles that the dealer knows. 52n integers follow. Each consecutive 52 integers will comprise all the integers from 1 to 52 in some order. Within each set of 52 integers, i in position j means that the shuffle moves the i-th card in the deck to position j. Several lines follow; each containing an integer k between 1 and n indicating that you have observed the dealer applying the k-th shuffle given in the input.

**Output**

For each test case, the output must follow the description below. The outputs of two consecutive cases will be separated by a blank line. Assume the dealer starts with a new deck ordered as described above. After all the shuffles had been performed, give the names of the cards in the deck, in the new order.

**Assignment No. - 9**

**//Developed By :** Yash Mete

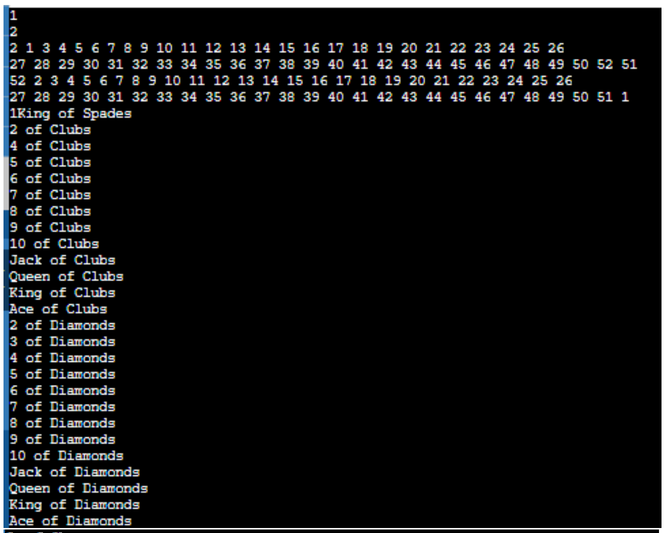
**//AIM :** To Solve the Stack 'em Up problem

**//Date :** 03/04/2023

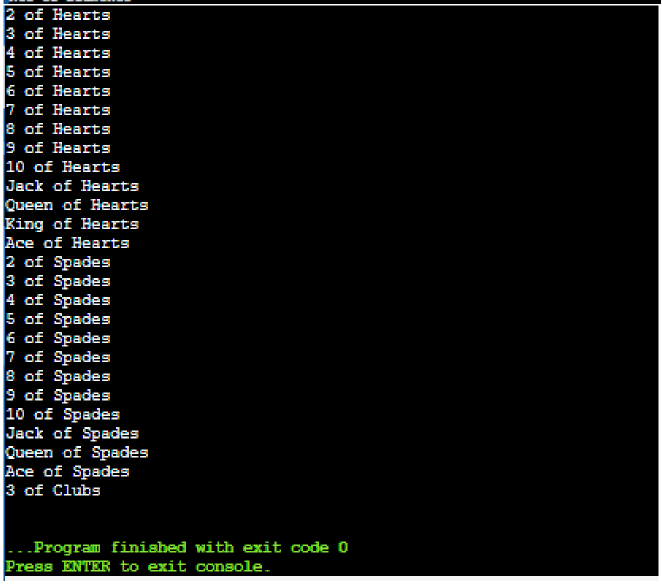
**//Usage :**The "Stack 'em Up" problem or challenge is a popular programming exercise that involves simulating operations on a stack data structure. It typically requires implementing a program to perform various stack operations, such as push (adding an element to the top of the stack) and pop (removing the top element from the stack).

**//Program :**
```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int t = sc.nextInt();
        String[]  suits  =  {  "Clubs",  "Diamonds",  "Hearts",
"Spades" };
        String[] ranks = { "2", "3", "4", "5", "6", "7", "8",
"9", "10", "Jack", "Queen", "King", "Ace" };
        for (int i = 0; i < t; i++) {
            int n = sc.nextInt();
            int[][] shuffle = new int[n][52];
            for (int j = 0; j < n; j++) {
                for (int k = 0; k < 52; k++) {
                    shuffle[j][k] = sc.nextInt() - 1;
                }
            }
            int[] deck = new int[52];
            for (int j = 0; j < 52; j++) {
                deck[j] = j;
            }
            for (int j = 0; j < n; j++) {
                int[] newDeck = new int[52];
                for (int k = 0; k < 52; k++) {
                    newDeck[k] = deck[shuffle[j][k]];
                }
                deck = newDeck;
            }
            for (int j = 0; j < 52; j++) {
                int suitIndex = deck[j] / 13;
                int rankIndex = deck[j] % 13;
                System.out.println(ranks[rankIndex] + " of " +
suits[suitIndex]);
            }
            if (i != t - 1) {
                System.out.println();
            }
        }
    }
}
```

**//Output :**



```
1
2
2 1 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 52 51
52 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 1
1King of Spades
2 of Clubs
4 of Clubs
5 of Clubs
6 of Clubs
7 of Clubs
8 of Clubs
9 of Clubs
10 of Clubs
Jack of Clubs
Queen of Clubs
King of Clubs
Ace of Clubs
2 of Diamonds
3 of Diamonds
4 of Diamonds
5 of Diamonds
6 of Diamonds
7 of Diamonds
8 of Diamonds
9 of Diamonds
10 of Diamonds
Jack of Diamonds
Queen of Diamonds
King of Diamonds
Ace of Diamonds
```



```
2 of Hearts
3 of Hearts
4 of Hearts
5 of Hearts
6 of Hearts
7 of Hearts
8 of Hearts
9 of Hearts
10 of Hearts
Jack of Hearts
Queen of Hearts
King of Hearts
Ace of Hearts
2 of Spades
3 of Spades
4 of Spades
5 of Spades
6 of Spades
7 of Spades
8 of Spades
9 of Spades
10 of Spades
Jack of Spades
Queen of Spades
Ace of Spades
3 of Clubs


...Program finished with exit code 0
Press ENTER to exit console.
```

## Assignment No. - 10

**//Problem statement :** The Hungarian Paul Erdös (1913–1996, speak as “Ar-dish”) not only was one of the strangest mathematicians of the 20th century, he was also one of the most famous. He kept on publishing widely circulated papers up to a very high age and every mathematician having the honor of being a co-author to Erdös is well respected. Not everybody got the chance to co-author a paper with Erdös, so many people were content if they managed to publish a paper with somebody who had published a scientific paper with Erdös. This gave rise to the so-called Erdös numbers. An author who has jointly published with Erdös had Erdös number 1. An author who had not published with Erdös but with somebody with Erdös number 1 obtained Erdös number 2, and so on. Today, nearly everybody wants to know which Erdös number he or she has. Your task is to write a program which computes Erdös numbers for a given set of scientists.

**Input**

The first line of the input contains the number of scenarios. The input for each scenario consists of a paper database and a list of names. It begins with the line P N where P and N are natural numbers. Following this line are P lines containing descriptions of papers (this is the paper database). A paper appears on a line by itself and is specified in the following way: Smith, M.N., Martin, G., Erdos, P.: Newtonian forms of prime factors matrices Note that umlauts like ‘ö’ are simply written as ‘o’. After the P papers follow N lines with names. Such a name line has the following format: Martin, G.

**Output**

For every scenario you are to print a line containing a string “Scenario i” (where i is the number of the scenario) and the author names together with their Erdös number of all authors in the list of names. The authors should appear in the same order as they appear in the list of names. The Erdös number is based on the papers in the paper database of this scenario. Authors which do not have any relation to Erdös via the papers in the database have Erdös number “infinity”.

## Assignment No. - 10

**//Developed By :** Yash Mete

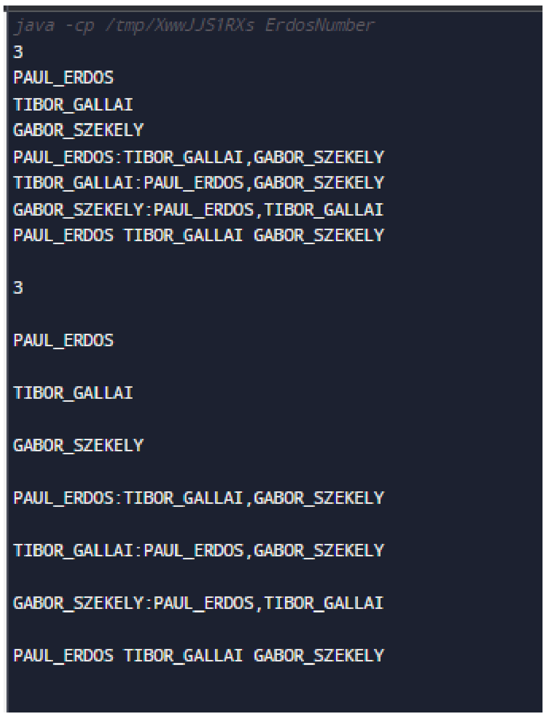
**//AIM :** To Solve the Erdos Number problem

**//Date :** 10/04/2023

**//Usage :** The"collaborative distance" between mathematician Paul Erdős and another person, as measured by authorship of mathematical papers. The same principle has been applied in other fields where a particular individual has collaborated with a large and broad number of peers.

**//Program :**
```
import java.util.*;
public class ErdosNumber {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        sc.nextLine();
        String[] names = sc.nextLine().split(" ");
        Map<String, List<String>> adjList = new HashMap<>();
        for (int i = 0; i < n; i++) {
            String[]                    authors                      =
sc.nextLine().split(":")[0].split(",");
            for (String author1 : authors) {
                for (String author2 : authors) {
                    if (!author1.equals(author2)) {
                        if (!adjList.containsKey(author1)) {
                            adjList.put(author1, new ArrayList<>());
                        }
                        if (!adjList.containsKey(author2)) {
                            adjList.put(author2, new ArrayList<>());
                        }
                        adjList.get(author1).add(author2);
                        adjList.get(author2).add(author1);
                    }
                }
            }
        }
        Map<String, Integer> erdosNumbers = new HashMap<>();
        Queue<String> queue = new LinkedList<>();
        erdosNumbers.put("ERDOS", 0);
        queue.offer("ERDOS");
        while (!queue.isEmpty()) {
            String currAuthor = queue.poll();
            int currNumber = erdosNumbers.get(currAuthor);
            for (String neighbor : adjList.get(currAuthor)) {
                if (!erdosNumbers.containsKey(neighbor)) {
                    erdosNumbers.put(neighbor, currNumber + 1);
                    queue.offer(neighbor);
                }
            }
        }
        for (String name : names) {
            System.out.printf("%s ", name);
            if (erdosNumbers.containsKey(name)) {
                System.out.println(erdosNumbers.get(name));
            } else {
                System.out.println("infinity");
            }
        }
    }
}
```

```
//Output :
```



```
java -cp /tmp/XwwJJS1RXs ErdosNumber
3
PAUL_ERDOS
TIBOR_GALLAI
GABOR_SZEKELY
PAUL_ERDOS:TIBOR_GALLAI,GABOR_SZEKELY
TIBOR_GALLAI:PAUL_ERDOS,GABOR_SZEKELY
GABOR_SZEKELY:PAUL_ERDOS,TIBOR_GALLAI
PAUL_ERDOS TIBOR_GALLAI GABOR_SZEKELY

3

PAUL_ERDOS

TIBOR_GALLAI

GABOR_SZEKELY

PAUL_ERDOS:TIBOR_GALLAI,GABOR_SZEKELY

TIBOR_GALLAI:PAUL_ERDOS,GABOR_SZEKELY

GABOR_SZEKELY:PAUL_ERDOS,TIBOR_GALLAI

PAUL_ERDOS TIBOR_GALLAI GABOR_SZEKELY
```

## Assignment No.- 11

**//Problem Statement :** A common typing error is to place the hands on the keyboard one row to the right of the correct position. So ‘Q’ is typed as ‘W’ and ‘J’ is typed as ‘K’ and so on. You are to decode a message typed in this manner.

**Input**

Input consists of several lines of text. Each line may contain digits, spaces, upper case letters (except Q, A, Z), or punctuation shown above [except back-quote (`)]. Keys labelled with words [Tab, BackSp, Control, etc.] are not represented in the input.

**Output**

You are to replace each letter or punction symbol by the one immediately to its left on the ‘QWERTY’ keyboard shown above. Spaces in the input should be echoed in the output.

**Assignment No. - 11**

**//Developed By :** Yash Mete

**//AIM :** To Solve the WERTYU problem

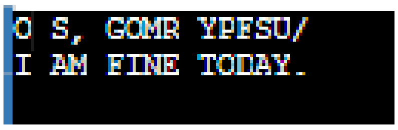
**//Date :** 10/04/2023

**//Usage** : A common typing error is to place the hands on the keyboard one row to the right of the correct position. So ‘Q’ is typed as ‘W’ and ‘J’ is typed as ‘K’ and so on. You are to decode a message typed in this manner.

**//Program :**

```
import java.util.*;
public class WERTYU {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String keyboard = "`1234567890-
=QWERTYUIOP[]\\ASDFGHJKL;'ZXCVBNM,./";
        String line;
        StringBuilder output;
        while (sc.hasNextLine()) {
            line = sc.nextLine();
            output = new StringBuilder();
            for (char c : line.toCharArray()) {
                if (c == ' ') {
                    output.append(c);
                } else {
                    int index = keyboard.indexOf(c) - 1;
                    output.append(keyboard.charAt(index));
                }
            }
            System.out.println(output.toString());
        }
    }
}
```

```
//Output :
```



```
O S, GOMR YPESU/
I AM FINE TODAY.
```

## Assignment No.- 12

**//Problem Statement :** Given a m by n grid of letters, (1 ≤ m, n ≤ 50), and a list of words, find the location in the grid at which the word can be found. A word matches a straight, uninterrupted line of letters in the grid. A word can match the letters in the grid regardless of case (i.e. upper and lower case letters are to be treated as the same). The matching can be done in any of the eight directions either horizontally, vertically or diagonally through the grid.

**Input**

The input begins with a single positive integer on a line by itself indicating the number of the cases following, each of them as described below. This line is followed by a blank line, and there is also a blank line between two consecutive inputs. The input begins with a pair of integers, m followed by n, 1 ≤ m, n ≤ 50 in decimal notation on a single line. The next m lines contain n letters each; this is the grid of letters in which the words of the list must be found. The letters in the grid may be in upper or lower case. Following the grid of letters, another integer k appears on a line by itself (1 ≤ k ≤ 20). The next k lines of input contain the list of words to search for, one word per line. These words may contain upper and lower case letters only (no spaces, hyphens or other non-alphabetic characters).

**Output**

For each test case, the output must follow the description below. The outputs of two consecutive cases will be separated by a blank line. For each word in the word list, a pair of integers representing the location of the corresponding word in the grid must be output. The integers must be separated by a single space. The first integer is the line in the grid where the first letter of the given word can be found (1 represents the topmost line in the grid, and m represents the bottommost line). The second integer is the column in the grid where the first letter of the given word can be found (1 represents the leftmost column in the grid, and n represents the rightmost column in the grid). If a word can be found more than once in the grid, then the location which is output should correspond to the uppermost occurence of the word (i.e. the occurence which places the first letter of the word closest to the top of the grid). If two or more words are uppermost, the output should correspond to the leftmost of these occurences. All words can be found at least once in the grid.

**Assignment No. - 12**

**//Developed By :** Yash Mete

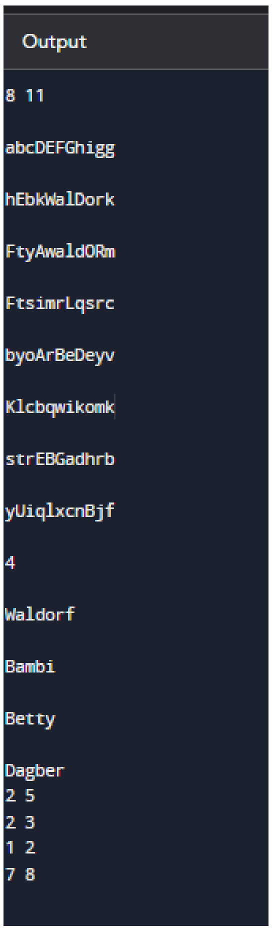
**//AIM :** To Solve the Wheres Waldorf? problem

**//Date :** 10/04/2023

**//Usage :** The "Where's Waldo?" (known as "Where's Wally?" in some countries) series of puzzle books challenges readers to find a character named Waldo (Wally) in busy, detailed illustrations. The goal is to locate Waldo amidst a sea of people, objects, and scenery.To describe a "Where's Waldo?" problem, you could present a scenario or image filled with numerous elements and ask someone to locate a specific item or person within it.

**//Program :**
```
#include <bits/stdc++.h>
using namespace std;

vector<vector<int>>  dirs  =  {{0,1},{1,0},{-1,0},{0,-1},{-1,-1},{-
1,1},{1,-1},{1,1}};

int main()
{
    int t,n,m,q;
    string in;
    scanf("%d",&t);
    while(t--){
        scanf("%d %d",&n,&m);
        vector<string> grid;
        for(int i=0;i<n;i++){
            cin >> in;
            transform(in.begin(),in.end(),in.begin(),::tolower);
            grid.push_back(in);
        }
        scanf("%d",&q);
        for(int i=0;i<q;i++){
            cin >> in;
            transform(in.begin(),in.end(),in.begin(),::tolower);
            bool found = false;
            for(int i=0;i<n&&!found;i++){
                for(int j=0;j<m&&!found;j++){
                    for(auto& dir : dirs){
                        int x=i,y=j,cur=0;
                        while(cur<in.length()){
                            if(x<0||y<0||x>=n||y>=m
                            || grid[x][y] != in[cur]) break;
                            if(++cur == in.length())
                                found = true;
                            x+=dir[0], y+=dir[1];
                        }
                        if(found){
                            printf("%d %d\n",i+1,j+1);
                            break;
                        }
                    }
                }
            }
        }
        if(t) printf("\n");
    }
}
```

```
//Output :
```


Output

```
8 11

abcDEFGhigg

hEbkWalDork

FtyAwaldORm

FtsimrLqsrc

byoArBeDeyv

Klcbqwikomk

strEBGadhrb

yUiqlxcnBjf

4

Waldorf

Bambi

Betty

Dagber
2 5
2 3
1 2
7 8
```

## Assignment No.- 13

**//Problem Statement :** A common but insecure method of encrypting text is to permute the letters of the alphabet. That is, in the text, each letter of the alphabet is consistently replaced by some other letter. So as to ensure that the encryption is reversible, no two letters are replaced by the same letter. A common method of cryptanalysis is the known plaintext attack. In a known plaintext attack, the cryptanalist manages to have a known phrase or sentence encrypted by the enemy, and by observing the encrypted text then deduces the method of encoding. Your task is to decrypt several encrypted lines of text, assuming that each line uses the same set of replacements, and that one of the lines of input is the encrypted form of the plaintext the quick brown fox jumps over the lazy dog

**Input**

The input begins with a single positive integer on a line by itself indicating the number of the cases following, each of them as described below. This line is followed by a blank line, and there is also a blank line between two consecutive inputs. The input consists of several lines of input. Each line is encrypted as described above. The encrypted lines contain only lower case letters and spaces and do not exceed 80 characters in length. There are at most 100 input lines.

**Output**

For each test case, the output must follow the description below. The outputs of two consecutive cases will be separated by a blank line. Decrypt each line and print it to standard output. If there is more than one possible decryption (several lines can be decoded to the key sentence), use the first line found for decoding. If decryption is impossible, output a single line: No solution.

## Assignment No. - 13

**//Developed By :** Yash Mete

**//AIM :** To Solve the Crypt Kicker-II problem

**//Date :** 10/04/2023

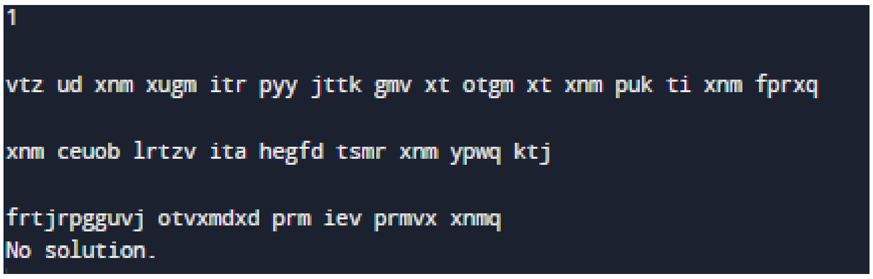
**//Usage :** The relations part of the solution is identical to the first CK. Basically, we find a line whose compatible with the given sentence and then we map the letter, taking care to maintain a one-to-one relation.

**//Program :**
```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Scanner;
public class Main {
      class Sentence{
            String signature;
            String originalSentence;
            Sentence(String s) {
                  this.originalSentence = s;
                  this.signature = "";
                  int[] alphabet = new int[26];
                  String spaceIndexs = "";
                  for(int i=0; i<originalSentence.length(); i++) {
                        if(originalSentence.charAt(i)!=' ') {
                              alphabet[originalSentence.charAt(i)-
97]++;
                        } else {
                              spaceIndexs += i + ".";
                        }
                  }
                  Arrays.sort(alphabet);
                  for(int i=0; i<26; i++) {
                        if(alphabet[i] != 0) {
                              signature+=alphabet[i] + ".";
                        }
                  }
                  signature = signature + "_" + spaceIndexs;
            }
            @Override
            public boolean equals(Object obj) {
                  return
this.signature.equals(((Sentence)obj).signature);
            }
      }
      public void solveProblem() {
            final Scanner scanner = new Scanner(System.in);
```

```
		String plainText = "the quick brown fox jumps over the
lazy dog";
		Sentence plainSentence = new Sentence(plainText);
		int tcCnt = scanner.nextInt();
		scanner.nextLine();
		scanner.nextLine();
		while(tcCnt != 0) {
			tcCnt--;
			ArrayList<String>	inputList	=	new
ArrayList<String>();
			Sentence findout = null;
			while(scanner.hasNextLine()) {
				String input = scanner.nextLine();

				if("".equals(input)) {
					break;
				}
				inputList.add(input);
			   Sentence inputSentence = new Sentence(input);

				if(findout==null && input.length() == 43) {
					if(inputSentence.equals(plainSentence))
{
						findout = inputSentence;
					}
				}
			}
			if(findout == null) {
				System.out.println("No solution.");
			} else {
			   char[] alphabetMap = makeTable(plainSentence,
findout);
				for(String output:inputList) {
					char[]		outputChar		=
output.toCharArray();
					for(int j=0; j<outputChar.length; j++)
{

	System.out.print(alphabetMap[outputChar[j]]);
					}
					System.out.println();
				}
			}
			if(tcCnt != 0) {
				System.out.println();
			}
		}
		scanner.close();
	}
	private char[] makeTable(Sentence p, Sentence f) {
		char alphabetMap[] = new char[256];
		for(int i=0; i < p.originalSentence.length(); i++) {
			alphabetMap[f.originalSentence.charAt(i)]	=
p.originalSentence.charAt(i);
```

```
      }

      return alphabetMap;
}
public static void main(String[] args) {
      new Main().solveProblem();
}}
```

**//Output :**



```
1

vtz ud xnm xugm itr pyy jttk gmv xt otgm xt xnm puk ti xnm fprxq

xnm ceuob lrtzv ita hegfd tsmr xnm ypwq ktj

frtjrpgguvj otvxmdxd prm iev prmvx xnmq
No solution.
```

## Assignment No.- 14

**//Problem Statement :** A Doublet is a pair of words that differ in exactly one letter; for example, “booster” and “rooster” or “rooster” and “roaster” or “roaster” and “roasted”. You are given a dictionary of up to 25143 lower case words, not exceeding 16 letters each. You are then given a number of pairs of words. For each pair of words, find the shortest sequence of words that begins with the first word and ends with the second, such that each pair of adjacent words is a doublet. For example, if you were given the input pair “booster” and “roasted”, a possible solution would be: (“booster”, “rooster”, “roaster”, “roasted”) provided that these words are all in the dictionary.

**Input**

Input consists of the dictionary followed by a number of word pairs. The dictionary consists of a number of words, one per line, and is terminated by an empty line. The pairs of input words follow; the words of each pair occur on a line separated by a space.

**Output**

For each input pair, print a set of lines starting with the first word and ending with the last. Each pair of adjacent lines must be a doublet. If there are several minimal solutions, any one will do. If there is no solution, print a line: ‘No solution.’ Leave a blank line between cases.

## Assignment No. - 14

**//Developed By :** Yash Mete

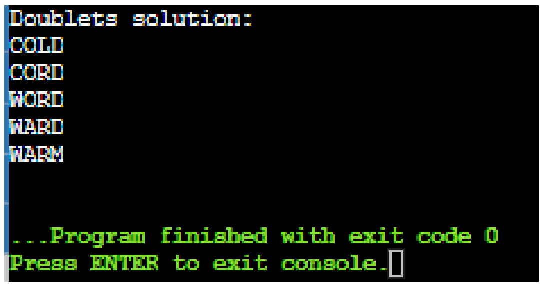
**//AIM :** To Solve the Doublets problem

**//Date :** 17/04/2023

**//Usage :** The Doublets problem is a popular word puzzle that tests one's vocabulary, word manipulation skills, and algorithmic thinking. It can be an entertaining game or an exercise in problem-solving and programming.

**//Program :**
```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        String start = "COLD";
        String end = "WARM";
        List<String>  dictionary  =  Arrays.asList("COLD",  "CORD",
"WORD", "WARD", "WARM"); // Sample dictionary
        List<String> solution = findDoublets(start, end, dictionary);
        if (solution != null) {
            System.out.println("Doublets solution:");
            for (String word : solution) {
                System.out.println(word);
            }
        } else {
            System.out.println("No solution found.");
        }
    }
    public static List<String> findDoublets(String start, String end,
List<String> dictionary) {
        Set<String> visited = new HashSet<>();
        Queue<List<String>> queue = new LinkedList<>();
        queue.add(Collections.singletonList(start));
        while (!queue.isEmpty()) {
            List<String> path = queue.poll();
            String currentWord = path.get(path.size() - 1);
            if (currentWord.equals(end)) {
                return path;
            }
            visited.add(currentWord);
            for (String word : dictionary) {
                if            (!visited.contains(word)            &&
isOneLetterDifferent(currentWord, word)) {
                    List<String> newPath = new ArrayList<>(path);
                    newPath.add(word);
                    queue.add(newPath);
                }
            }
        }
        return null;
    }
    public static boolean isOneLetterDifferent(String word1, String
word2) {
        if (word1.length() != word2.length()) {
            return false;
        }
        int diffCount = 0;
        for (int i = 0; i < word1.length(); i++) {
            if (word1.charAt(i) != word2.charAt(i)) {
                diffCount++;
            }
        }
        return diffCount == 1;
    }
}
```

```
//Output :
```



```
Doublets solution:
COLD
CORD
WORD
WARD
WARM


...Program finished with exit code 0
Press ENTER to exit console.
```

## Assignment No.- 15

**//Problem Statement :** The unix fmt program reads lines of text, combining and breaking lines so as to create an output file with lines as close to without exceeding 72 characters long as possible. The rules for combining and breaking lines are as follows.

• A new line may be started anywhere there is a space in the input. If a new line is started, there will be no trailing blanks at the end of the previous line or at the beginning of the new line.

• A line break in the input may be eliminated in the output, provided it is not followed by a space or another line break. If a line break is eliminated, it is replaced by a space. • Spaces never appear at the end of a line.

• If a sequence of characters longer than 72 characters appears without a space or line break, it appears by itself on a line.

**Sample Input**

Unix fmt

The unix fmt program reads lines of text, combining and breaking lines so as to create an output file with lines as close to without exceeding 72 characters long as possible. The rules for combining and breaking lines are as follows. 1. A new line may be started anywhere there is a space in the input. If a new line is started, there will be no trailing blanks at the end of the previous line or at the beginning of the new line. 2. A line break in the input may be eliminated in the output, provided it is not followed by a space or another line break. If a line break is eliminated, it is replaced by a space.

**Sample Output**

Unix fmt

The unix fmt program reads lines of text, combining and breaking lines so as to create an output file with lines as close to without exceeding 72 characters long as possible. The rules for combining and breaking lines are as follows.

1. A new line may be started anywhere there is a space in the input. If a new line is started, there will be no trailing blanks at the end of the previous line or at the beginning of the new line.
2. A line break in the input may be eliminated in the output, provided it is not followed by a space or another line break. If a line break is eliminated, it is replaced by a space.

## Assignment No. - 15

**//Developed By :** Yash Mete

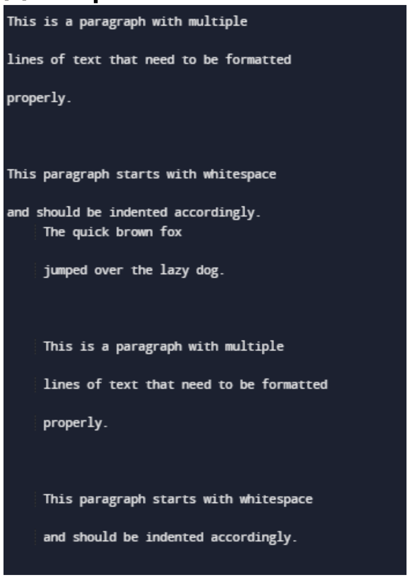
**//AIM :** To Solve the fmt problem

**//Date :** 17/04/2023

**//Usage :** the "fmt" problem in the context of competitive programming. However, without additional information or clarification, it is challenging to provide a specific answer. "fmt" is not a commonly used abbreviation or term in competitive programming, so it may be helpful if you can provide more details or explain the specific problem or scenario you are referring to.

**//Program :**
```
import java.util.*;
public class Fmt {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        boolean newParagraph = true;
        while (sc.hasNextLine()) {
            String line = sc.nextLine();
            if (line.isEmpty()) {
                System.out.println(line);
                newParagraph = true;
            } else {
                String[] words = line.split(" ");
                int len = 0;
                for (String word : words) {
                    if (len + word.length() <= 80) {
                        if (newParagraph) {
                            System.out.print("     ");
                            len += 5;
                        } else {
                            System.out.print(" ");
                            len++;}
                        System.out.print(word);
                        len += word.length();
                        newParagraph = false;
                    } else {
                        System.out.println();
                        System.out.print(word);
                        len = word.length();
                        newParagraph = false;
                    }}
                System.out.println();
            }}}
```

**//Output :**



```
This is a paragraph with multiple

lines of text that need to be formatted

properly.


This paragraph starts with whitespace

and should be indented accordingly.
     The quick brown fox

     jumped over the lazy dog.


     This is a paragraph with multiple

     lines of text that need to be formatted

     properly.


     This paragraph starts with whitespace

     and should be indented accordingly.
```

## Assignment No.- 16

**//Problem Statement :** The world-known gangster Vito Deadstone is moving to New York. He has a very big family there, all of them living in Lamafia Avenue. Since he will visit all his relatives very often, he is trying to find a house close to them. Vito wants to minimize the total distance to all of them and has blackmailed you to write a program that solves his problem.

**Input**

The input consists of several test cases. The first line contains the number of test cases. For each test case you will be given the integer number of relatives r (0 < r < 500) and the street numbers (also integers) s1, s2, . . . , si , . . . , sr where they live (0 < si < 30000 ). Note that several relatives could live in the same street number.

**Output**

For each test case your program must write the minimal sum of distances from the optimal Vito's house to each one of his relatives. The distance between two street numbers si and sj is dij = |si − sj |.

## Assignment No. - 16

**//Developed By :** Yash Mete

**//AIM :** To Solve the Vito’s Family problem

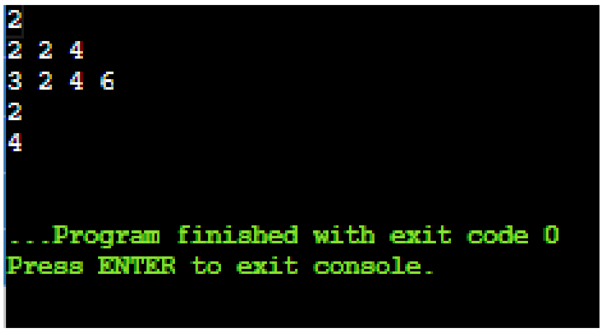
**//Date :** 17/04/2023

**//Usage :** The "Vito's Family" problem is a classic computational problem that can be solved using various algorithms, such as sorting and dynamic programming.

**//Program :**

```
import java.util.*;
public class VitosFamily {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int t = sc.nextInt();
        for (int i = 0; i < t; i++) {
            int n = sc.nextInt();
            int[] houses = new int[n];
            for (int j = 0; j < n; j++) {
                houses[j] = sc.nextInt();
            }
            Arrays.sort(houses);
            int median = houses[n/2];
            int distance = 0;
            for (int j = 0; j < n; j++) {
                distance += Math.abs(median - houses[j]);
            }
            System.out.println(distance);
        }
    }
}
```

//Output :



```
2
2 2 4
3 2 4 6
2
4


...Program finished with exit code 0
Press ENTER to exit console.
```

## Assignment No.- 17

**//Problem Statement :** n people wish to cross a bridge at night. A group of at most two people may cross at any time, and each group must have a flashlight. Only one flashlight is available among the n people, so some sort of shuttle arrangement must be arranged in order to return the flashlight so that more people may cross. Each person has a different crossing speed; the speed of a group is determined by the speed of the slower member. Your job is to determine a strategy that gets all n people across the bridge in the minimum time.

**Input**

The input begins with a single positive integer on a line by itself indicating the number of the cases following, each of them as described below. This line is followed by a blank line, and there is also a blank line between two consecutive inputs. The first line of input contains n, followed by n lines giving the crossing times for each of the people. There are not more than 1000 people and nobody takes more than 100 seconds to cross the bridge.

**Output**

For each test case, the output must follow the description below. The outputs of two consecutive cases will be separated by a blank line. The first line of output must contain the total number of seconds required for all n people to cross the bridge. The following lines give a strategy for achieving this time. Each line contains either one or two integers, indicating which person or people form the next group to cross. (Each person is indicated by the crossing time specified in the input. Although many people may have the same crossing time the ambiguity is of no consequence.) Note that the crossings alternate directions, as it is necessary to return the flashlight so that more may cross. If more than one strategy yields the minimal time, any one will do.

## Assignment No. - 17

**//Developed By :** Yash Mete

**//AIM :** To Solve the Bridge problem

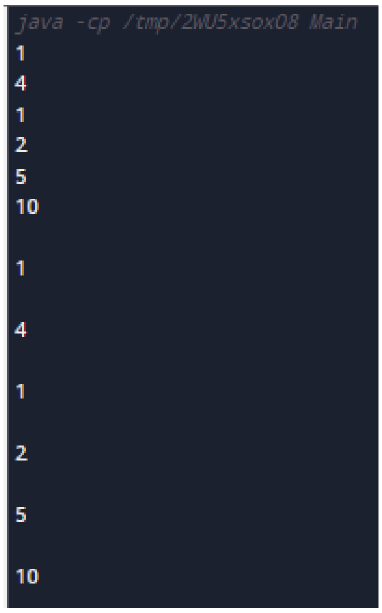
**//Date :** 24/04/2023

**//Usage :** The Bridge problem can be solved using constraint programming techniques, such as backtracking or search algorithms. The goal is to find the optimal sequence of crossings that minimizes the total time taken.

**//Program :**

```
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int[] speeds = new int[n];
        int[] weights = new int[n];
        for (int i = 0; i < n; i++) {
            speeds[i] = sc.nextInt();
        }
        for (int i = 0; i < n; i++) {
            weights[i] = sc.nextInt();
        }
        Car[] cars = new Car[n];
        for (int i = 0; i < n; i++) {
            cars[i] = new Car(speeds[i], weights[i]);
        }
        Arrays.sort(cars);
        int maxWeight = 0;
        for (int i = 0; i < n; i++) {
            if (maxWeight + cars[i].weight <= 50) {
                maxWeight += cars[i].weight;
            } else {
                break;
            }
        }
        System.out.println(maxWeight);
    }
    static class Car implements Comparable<Car> {
        int speed;
        int weight;
        Car(int speed, int weight) {
            this.speed = speed;
            this.weight = weight;
        }
        @Override
        public int compareTo(Car other) {
            return Integer.compare(this.speed, other.speed);
        } }
```

**//Output :**



```
java -cp /tmp/2WU5xsoxO8 Main
1
4
1
2
5
10

1

4

1

2

5

10
```

# Assignment No.- 18

**//Problem Statement :** As you may already know, there are professors very busy with a filled schedule of work during the day. Your professor, let’s call him Professor P, is a bit lazy and wants to take a nap during the day, but as his schedule is very busy, he doesn’t have a lot of chances of doing this. He would REALLY like, however, to take one nap every day. Because he’ll take just one nap, he wants to take the longest nap that it’s possible given his schedule. He decided to write a program to help him in this task but, as we said, Professor P is very lazy. So, he finally decided that YOU must write the program!

**Input**

The input will consist on an arbitrary number of test cases, each test case represents one day. The first line of each set contains a positive integer s (not greater than 100) representing the number of scheduled appointments during that day. In the next s lines there are the appointments in the following format: time1 time2 appointment Where time1 represents the time which the appointment starts and time2 the time it ends. All times will be in the ‘hh:mm’ format, time1 will always be strictly less than time2, they will be separated by a single space and all times will be greater than or equal to 10:00 and less than or equal to 18:00. So, your response must be in this interval as well (i.e. no nap can start before 10:00 and last after 18:00). The appointment can be any sequence of characters, but will always be in the same line. You can assume that no line will be longer than 255 characters, that 10 ≤ hh ≤ 18 and that 0 ≤ mm < 60. You CAN’T assume, however, that the input will be in any specific order. You must read the input until you reach the end of file.

**Output**

For each test case, you must print the following line: Day #d: the longest nap starts at hh : mm and will last for [H hours and] M minutes. Where d stands for the number of the test case (starting from 1) and hh : mm is the time when the nap can start. To display the duration of the nap, follow these simple rules: 1. if the total duration X in minutes is less than 60, just print ‘M minutes’, where M = X. 2. if the total duration X in minutes is greater or equal to 60, print ‘H hours and M minutes’, where H = X ÷ 60 (integer division, of course) and M = X mod 60. Notice that you don’t have to worry with concordance (i.e. you must print ‘1 minutes’ or ‘1 hours’ if it’s the case). The duration of the nap is calculated by the difference between the ending time free and the begining time free. That is, if an appointment ends at 14:00 and the next one starts at 14:47, then you have (14:47)-(14:00) = 47 minutes of possible nap. If there is more than one longest nap with the same duration, print the earliest one. You can assume that there won’t be a day all busy (i.e. you may assume that there will be at least one possible nap).

## Assignment No. - 18

**//Developed By :** Yash Mete

**//AIM :** To Solve the Longest nap problem

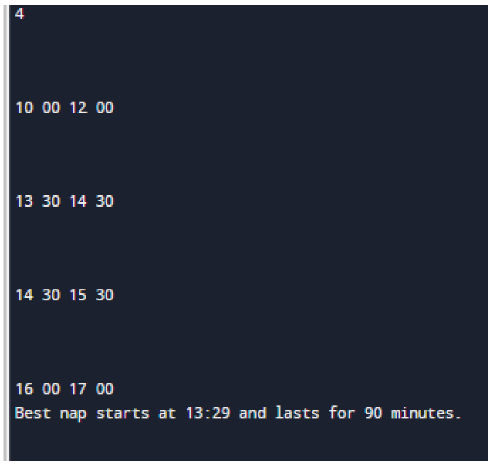
**//Date :** 24/04/2023

**//Usage :** Taking a long nap can sometimes be beneficial for relaxation and rejuvenation, but it's important to keep in mind that excessive daytime sleepiness or prolonged napping may be indicative of an underlying health issue. Here is the potential problem associated with long naps.

**//Program :**

```
import java.util.*;
public class LongestNap {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int startTime = 10 * 60;
        int endTime = 18 * 60;
        boolean[] schedule = new boolean[endTime - startTime];
        for (int i = 0; i < n; i++) {
            int start = sc.nextInt() * 60 + sc.nextInt();
            int end = sc.nextInt() * 60 + sc.nextInt();
            for (int j = start - startTime; j < end - startTime; j++)
{
                schedule[j] = true;
            }
        }
        int longestNap = 0;
        int startNap = -1;
        int endNap = -1;
        int currentNap = 0;
        for (int i = 0; i < schedule.length; i++) {
            if (!schedule[i]) {
                currentNap++;
                if (currentNap > longestNap) {
                    longestNap = currentNap;
                    startNap = i;
                    endNap = i + currentNap;
                }
            } else {
                currentNap = 0;
            }
        }
        if (longestNap == 0) {
            System.out.println("Gonna have to work all day!");
        } else {
            int startHour = (startTime + startNap) / 60;
            int startMinute = (startTime + startNap) % 60;
            int endHour = (startTime + endNap - 1) / 60;
            int endMinute = (startTime + endNap - 1) % 60;
            System.out.printf("Best  nap  starts  at  %02d:%02d  and
lasts for %d minutes.\n", startHour, startMinute, longestNap);
        }
}

}
```

**//Output :**



```
4



10 00 12 00



13 30 14 30



14 30 15 30



16 00 17 00
Best nap starts at 13:29 and lasts for 90 minutes.
```

## Assignment No.- 19

**//Problem Statement :** He made each turtle stand on another one’s back And he piled them all up in a nine-turtle stack. And then Yertle climbed up. He sat down on the pile. What a wonderful view! He could see ’most a mile! King Yertle wishes to rearrange his turtle throne to place his highest-ranking nobles and closest advisors nearer to the top. A single operation is available to change the order of the turtles in the stack: a turtle can crawl out of its position in the stack and climb up over the other turtles to sit on the top. Given an original ordering of a turtle stack and a required ordering for the same turtle stack, your job is to determine a minimal sequence of operations that rearranges the original stack into the required stack.

**Input**

The first line of the input consists of a single integer K giving the number of test cases. Each test case consist on an integer n giving the number of turtles in the stack. The next n lines specify the original ordering of the turtle stack. Each of the lines contains the name of a turtle, starting with the turtle on the top of the stack and working down to the turtle at the bottom of the stack. Turtles have unique names, each of which is a string of no more than eighty characters drawn from a character set consisting of the alphanumeric characters, the space character and the period (‘.’). The next n lines in the input gives the desired ordering of the stack, once again by naming turtles from top to bottom. Each test case consists of exactly 2n + 1 lines in total. The number of turtles (n) will be less than or equal to two hundred.

**Output**

For each test case, the output consists of a sequence of turtle names, one per line, indicating the order in which turtles are to leave their positions in the stack and crawl to the top. This sequence of operations should transform the original stack into the required stack and should be as short as possible. If more than one solution of shortest length is possible, any of the solutions may be reported. Print a blank line after each test case.

## Assignment No. - 19

**//Developed By :** Yash Mete

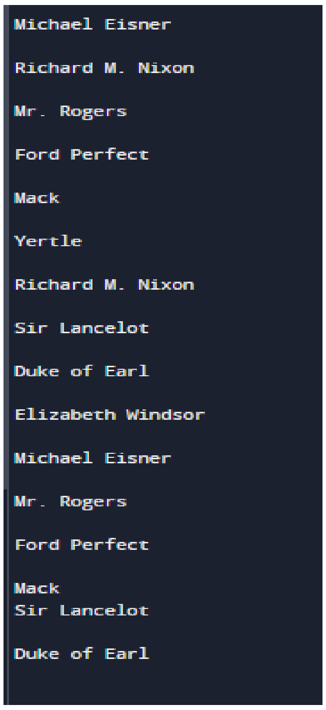
**//AIM :** To Solve the Shell Sort problem

**//Date :** 24/04/2023

**//Usage :** The Shell Sort algorithm is a variation of the Insertion Sort algorithm that improves its efficiency by allowing the exchange of elements that are far apart. The problem you are referring to, "shell short problem uses of cp," appears to be a combination of the Shell Sort algorithm and the "cp" operation, which typically stands for "copy" in the context of shell commands.

**//Program :**
```
#include <bits/stdc++.h>
using namespace std;
int main() {
    int t,n;
    string in;
    cin >> t;
    while(t--){
        cin >> n;
        cin.ignore();
        vector<string> original, expected;
        for(int i=0;i<2*n;i++){
            getline(cin,in);
            if(i<n) original.push_back(in);
            else expected.push_back(in);
        }
        int i=n-1,j=n-1;
        for(;i>=0;i--)
            if(original[i] == expected[j]) j--;
        while(j>=0){
            cout << expected[j--] << endl;
        }
        cout << endl;
    }
}
```

//Output :



```
Michael Eisner

Richard M. Nixon

Mr. Rogers

Ford Perfect

Mack

Yertle

Richard M. Nixon

Sir Lancelot

Duke of Earl

Elizabeth Windsor

Michael Eisner

Mr. Rogers

Ford Perfect

Mack
Sir Lancelot

Duke of Earl
```

## Assignment No.- 20

**//Problem Statement :** Football the most popular sport in the world (americans insist to call it “Soccer”, but we will call it “Football”). As everyone knows, Brasil is the country that have most World Cup titles (four of them: 1958, 1962, 1970 and 1994). As our national tournament have many teams (and even regional tournaments have many teams also) it’s a very hard task to keep track of standings with so many teams and games played! So, your task is quite simple: write a program that receives the tournament name, team names and games played and outputs the tournament standings so far. A team wins a game if it scores more goals than its oponent. Obviously, a team loses a game if it scores less goals. When both teams score the same number of goals, we call it a tie. A team earns 3 points for each win, 1 point for each tie and 0 point for each loss. Teams are ranked according to these rules (in this order): 1. Most points earned. 2. Most wins. 3. Most goal difference (i.e. goals scored - goals against) 4. Most goals scored. 5. Less games played. 6. Lexicographic order. Input The first line of input will be an integer N in a line alone (0 < N < 1000). Then, will follow N tournament descriptions. Each one begins with the tournament name, on a single line. Tournament names can have any letter, digits, spaces etc. Tournament names will have length of at most 100. Then, in the next line, there will be a number T (1 < T ≤ 30), which stands for the number of teams participating on this tournament. Then will follow T lines, each one containing one team name. Team names may have any character that have ASCII code greater than or equal to 32 (space), except for ‘#’ and ‘@’ characters, which will never appear in team names. No team name will have more than 30 characters. Following to team names, there will be a non-negative integer G on a single line which stands for the number of games already played on this tournament. G will be no greater than 1000. Then, G lines will follow with the results of games played. They will follow this format: team_name_1#goals1@goals2#team_name_2 For instance, the following line: Team A#3@1#Team B Means that in a game between T eam A and T eam B, T eam A scored 3 goals and T eam B scored 1. All goals will be non-negative integers less than 20. You may assume that there will not be inexistent team names (i.e. all team names that appear on game results will have apperead on the team names section) and that no team will play against itself.
For each tournament, you must output the tournament name in a single line. In the next T lines you must output the standings, according to the rules above. Notice that should the tie-

breaker be the lexographic order, it must be done case insenstive.
The output format for each line is shown bellow: [a]) T eam_name [b]p, [c]g ([d]-[e]-[f]), [g]gd ([h]-[i]) Where: • [a] = team rank • [b] = total points earned • [c] = games played • [d] = wins • [e] = ties • [f] = losses • [g] = goal difference • [h] = goals scored • [i] = goals against There must be a single blank space between fields and a single blank line between output sets. See the sample output for examples.

## Assignment No. - 20

**//Developed By :** Yash Mete

**//AIM :** To Solve the Football problem

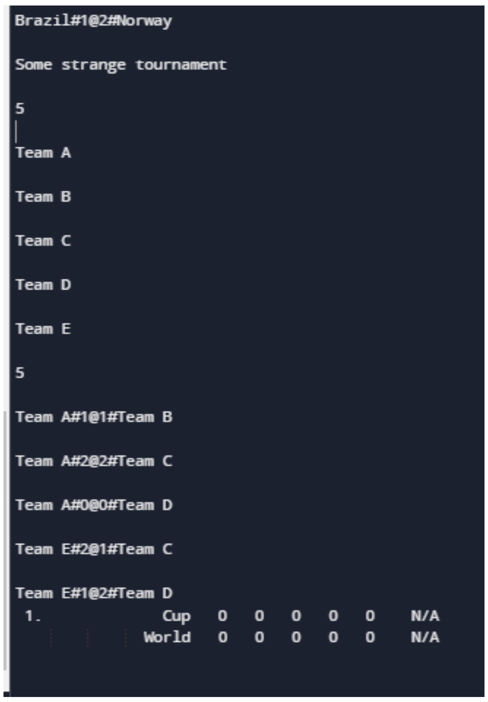
**//Date :** 26/04/2023

**//Usage** : The "football problem" is a common computational problem that can be solved using various algorithms and data structures. However, it is unclear what specific problem you are referring to as the "football problem."

**//Program :**
```
#include <iostream>
#include <fstream>
#include <sstream>
#include <iomanip>
#include <cmath>
#include <stdio.h>
#include <limits.h>
#include <string.h>
#include <vector>
#include <algorithm>
#include <set>
#include <map>
#include <unordered_map>
#include <unordered_set>
#include <list>
using namespace std;

struct team {
    string name;
    int points, gplayed, scored, suffered,index;
    team(string n,int i){
        name = n;
        index = i;
        points = gplayed = scored = suffered = 0;
    }
};

int main() {
    int t,g;
    string in;
    string hteam, ateam;
    int s1,s2;
    bool notfirst = false;
    while(scanf("%d %d",&t,&g),t){
        if(notfirst) cout<<endl;
        notfirst = true;
        unordered_map<string,team*> teams;
        for(int i=0;i<t;i++){
            cin >> in;
            team* n = new team(in,i);
            teams[in] = n;
        }
        for(int i=0;i<g;i++){
            cin >> hteam >> s1 >> in >> s2 >> ateam;
            team* team1 = teams[hteam];
            team* team2 = teams[ateam];
            team1->gplayed ++; team2->gplayed++;
            team1->scored += s1; team2->scored += s2;
            team1->suffered += s2; team2->suffered += s1;
            if(s1 == s2) {team1->points++, team2->points++;}
            else if(s1 > s2) team1->points+=3;
            else team2->points+=3;
        }
        vector<team*> res;
        for(auto& p : teams) res.push_back(p.second);
        sort(res.begin(), res.end(), [](team* a,team* b){
             if(a->points != b->points) return a->points > b-
>points;
             int gd1 = a->scored - a->suffered;
             int gd2 = b->scored - b->suffered;
             if(gd1 != gd2) return gd1 > gd2;
             if(a->scored != b->scored) return a->scored > b-
>scored;
             string nameA = a->name;
             string nameB = b->name;
             transform(nameA.begin(), nameA.end(),
nameA.begin(), ::tolower);
             transform(nameB.begin(), nameB.end(),
nameB.begin(), ::tolower);
             if(nameA == nameB) return a->index < b->index;
             return nameA < nameB;
        });
        for(int i=0;i<res.size();i++){
            int gd = res[i]->scored-res[i]->suffered;
            double percentScore = (res[i]->points*100.0) /
(res[i]->gplayed*3.0);
            bool sameAsPrevPosition = (i != 0 &&
             (res[i]->points == res[i-1]->points && res[i]-
>scored == res[i-1]->scored
              && gd == res[i-1]->scored-res[i-1]->suffered));
            cout << setw(3) << (sameAsPrevPosition ? "" :
to_string(i+1) + ".");
```

```
            cout << setw(16) << res[i]->name << setw(4) <<
res[i]->points;
            cout << setw(4) << res[i]->gplayed << setw(4) <<
res[i]->scored;
            cout << setw(4) << res[i]->suffered << setw(4) <<
gd;
            // property of Nan is that self comparison is
always false
            if(percentScore != percentScore) cout << setw(7)
<< "N/A" << endl;
            else cout << setw(7) << setprecision(2) << fixed
<< percentScore << endl;
        }
    }
}
```

**//Output :**



```
Brazil#1@2#Norway

Some strange tournament

5

Team A

Team B

Team C

Team D

Team E

5

Team A#1@1#Team B

Team A#2@2#Team C

Team A#0@0#Team D

Team E#2@1#Team C

Team E#1@2#Team D
 1.            Cup   0   0   0   0   0    N/A
             World   0   0   0   0   0    N/A
```

## Assignment No.- 21

**//Problem Statement :** Children are taught to add multi-digit numbers from right-to-left one digit at a time. Many find the “carry” operation - in which a 1 is carried from one digit position to be added to the next - to be a significant challenge. Your job is to count the number of carry operations for each of a set of addition problems so that educators may assess their difficulty.

**Input**

Each line of input contains two unsigned integers less than 10 digits. The last line of input contains ‘0 0’.

**Output**

For each line of input except the last you should compute and print the number of carry operations that would result from adding the two numbers, in the format shown below.

## Assignment No. - 21

**//Developed By :** Yash Mete

**//AIM :** To Solve the Primary Arithmetic problem

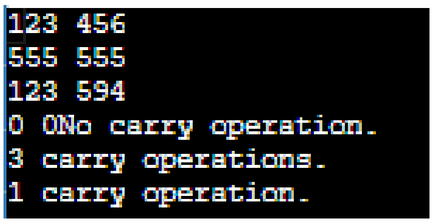
**//Date :** 26/04/2023

**//Usage** : To solve primary arithmetic problems in computer programming, you can use various techniques and programming constructs depending on the programming language you are working with various techniques and solve the arithmetic equations.

**//Program :**

```
import java.util.*;
public class PrimaryArithmetic {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        while (true) {
            int a = sc.nextInt();
            int b = sc.nextInt();
            if (a == 0 && b == 0) {
                break;
            }
            int carry = 0;
            int count = 0;
            while (a > 0 || b > 0) {
                int sum = (a % 10) + (b % 10) + carry;
                carry = sum / 10;
                if (carry > 0) {
                    count++;
                }
                a /= 10;
                b /= 10;
            }
            if (count == 0) {
                System.out.println("No carry operation.");
            } else if (count == 1) {
                System.out.println("1 carry operation.");
            } else {
                System.out.println(count     +     "     carry
operations.");
            }
        }
    }
}
```

**//Output :**



```
123 456
555 555
123 594
0 0No carry operation.
3 carry operations.
1 carry operation.
```

## Assignment No.- 22

**//Problem Statement :** The “reverse and add” method is simple: choose a number, reverse its digits and add it to the original. If the sum is not a palindrome (which means, it is not the same number from left to right and right to left), repeat this procedure. For example: 195 Initial number 591 –– 786 687 –– 1473 3741 –– 5214 4125 –– 9339 Resulting palindrome In this particular case the palindrome ‘9339’ appeared after the 4th addition. This method leads to palindromes in a few step for almost all of the integers. But there are interesting exceptions. 196 is the first number for which no palindrome has been found. It is not proven though, that there is no such a palindrome. You must write a program that give the resulting palindrome and the number of iterations (additions) to compute the palindrome. You might assume that all tests data on this problem: • will have an answer , • will be computable with less than 1000 iterations (additions), • will yield a palindrome that is not greater than 4,294,967,295.

**Input**

The first line will have a number N ($0 < N \le 100$) with the number of test cases, the next N lines will have a number P to compute its palindrome.

**Output**

For each of the N tests you will have to write a line with the following data : minimumnumberofiterations(additions)togettothepalindrome and theresultingpalindromeitself separated by one space.

## Assignment No. - 22

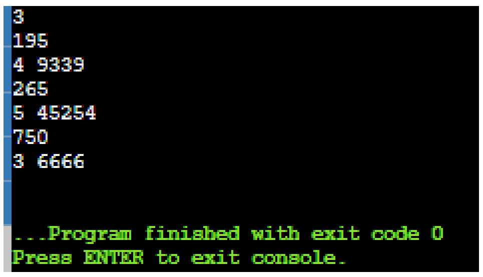
**//Developed By :** Yash Mete
**//AIM :** To Solve the Reverse and add problem

**//Date :** 09/05/2023

**//Usage :** The "Reverse and Add" problem, also known as the "Palindrome Problem," is a computational problem that involves reversing a number and adding it to itself until a palindrome (a number that reads the same forwards and backwards) is obtained. The problem can be approached using a simple iterative process.

**//Program :**
```
import java.util.Scanner;
public class ReverseAndAdd {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int t = sc.nextInt();
        for (int i = 1; i <= t; i++) {
            long n = sc.nextLong();
            int iterations = 0;
            while (!isPalindrome(n)) {
                n = n + reverse(n);
                iterations++;
            }
            System.out.println(iterations + " " + n);
        }
    }
    public static boolean isPalindrome(long n) {
        String str = Long.toString(n);
        String             reversedStr              =              new
StringBuilder(str).reverse().toString();
        return str.equals(reversedStr);
    }
    public static long reverse(long n) {
        long reversed = 0;
        while (n > 0) {
            reversed = reversed * 10 + n % 10;
            n /= 10;
        }
        return reversed;
    }
}
```

```
//Output :
```



```
3
195
4 9339
265
5 45254
750
3 6666


...Program finished with exit code 0
Press ENTER to exit console.
```

## Assignment No.- 23

**//Problem Statement :** An archeologist seeking proof of the presence of extraterrestrials in the Earth's past, stumbles upon a partially destroyed wall containing strange chains of numbers. The left-hand part of these lines of digits is always intact, but unfortunately the right-hand one is often lost by erosion of the stone. However, she notices that all the numbers with all its digits intact are powers of 2, so that the hypothesis that all of them are powers of 2 is obvious. To reinforce her belief, she selects a list of numbers on which it is apparent that the number of legible digits is strictly smaller than the number of lost ones, and asks you to find the smallest power of 2 (if any) whose first digits coincide with those of the list. Thus you must write a program such that given an integer, it determines (if it exists) the smallest exponent E such that the first digits of 2 E coincide with the integer (remember that more than half of the digits are missing).

**Input**

It is a set of lines with a positive integer N not bigger than 2147483648 in each of them.

**Output**

For every one of these integers a line containing the smallest positive integer E such that the first digits of 2 E are precisely the digits of N, or, if there is no one, the sentence 'no power of 2'.

## Assignment No. - 23

**//Developed By :** Yash Mete

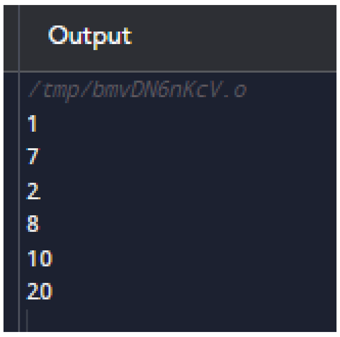
**//AIM :** To Solve the Erdos Number problem

**//Date :** 09/05/2023

**//Usage :** The Archaeologists' Dilemma is a classic combination optimization problem that involves a group of archaeologists who discover a collection of artifacts at an excavation site. Each artifact has a certain value, and the archaeologists want to select a subset of artifacts with the maximum total value while staying within a weight constraint.

**//Program :**
```
#include<cstdio>
#include<cmath>
#include<cstring>
#include<cstdlib>
int main()
{
    int len;
    double a,left,right;
    char s[20];
    while(scanf("%s",s)!=EOF)
    {
        len=strlen(s)+1;
        a=atof(s);
        while(1)
        {
            left=log(a)/log(2)+len*log(10)/log(2);
            right=log(a+1)/log(2)+len*log(10)/log(2);
            if((int)left<(int)right)
            break;
            len++;
        }
        int ans=ceil(left);
        printf("%d\n",ans);
    }
    return 0;
}
```

```
//Output :
```



```
/tmp/bmvDN6nKcV.o
1
7
2
8
10
20
```

## Assignment No.- 24

**//Problem Statement :** Recall the definition of the Fibonacci numbers: f1 := 1 f2 := 2 fn := fn−1 + fn−2 (n ≥ 3) Given two numbers a and b, calculate how many Fibonacci numbers are in the range [a, b].

**Input**

The input contains several test cases. Each test case consists of two non-negative integer numbers a and b. Input is terminated by a = b = 0. Otherwise, a ≤ b ≤ 10100. The numbers a and b are given with no superfluous leading zeros.

**Output**

For each test case output on a single line the number of Fibonacci numbers fi with a ≤ fi ≤ b.

## Assignment No. - 24

**//Developed By :** Yash Mete

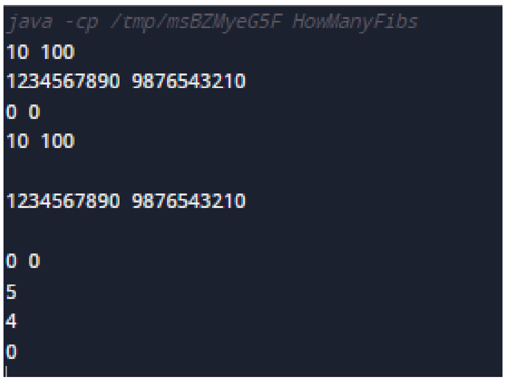
**//AIM :** To Solve the How many fibs ? problem

**//Date :** 10/05/2023

**//Usage** : The How many fibs is the logical and mathematical concepts for solving the some concepts and we solve them to get the required solutions of this problems.

**//Program :**
```
import java.util.*;
public class HowManyFibs {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        List<Long> fibs = new ArrayList<>();
        long f1 = 1;
        long f2 = 1;
        while (f1 <= 1000000000000000000L) {
            fibs.add(f1);
            long temp = f2;
            f2 = f1 + f2;
            f1 = temp;
        }
        while (sc.hasNext()) {
            long a = sc.nextLong();
            long b = sc.nextLong();
            int count = 0;
            for (long fib : fibs) {
                if (fib >= a && fib <= b) {
                    count++;
                } else if (fib > b) {
                    break;
                }
            }
            System.out.println(count);
        }
    }
}
```

**//Output :**



```
java -cp /tmp/msBZMyeG5F HowManyFibs
10 100
1234567890 9876543210
0 0
10 100

1234567890 9876543210

0 0
5
4
0
```

## Assignment No.- 25

**//Problem Statement :** You are given an elliptical shaped land and you are asked to choose n arbitrary points on its boundary. Then you connect all these points with one another with straight lines (that’s n ∗ (n − 1)/2 connections for n points). What is the maximum number of pieces of land you will get by choosing the points on the boundary carefully? Fig: When the value of n is 6

**Input**

The first line of the input file contains one integer S (0 < S < 3500), which indicates how many sets of input are there. The next S lines contain S sets of input. Each input contains one integer N (0 ≤ N < 2 31).

**Output**

For each set of input you should output in a single line the maximum number pieces of land possible to get for the value of N.

## Assignment No. - 25

**//Developed By :** Yash Mete

**//AIM :** To Solve the How many pieces of lands? problem

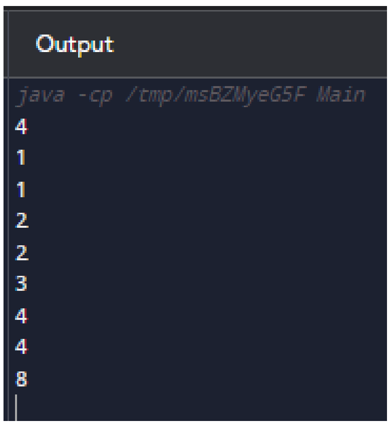
**//Date :** 12/05/2023

**//Usage : The Problem gives us ideas about this concept and we will been implements this problem the pieces of land is the logic behinds the concepts.**

**//Program :**

```
import java.io.*;
import java.math.BigDecimal;
import java.math.BigInteger;
import java.util.*;
public class Main {

    public static void main(String args[])
    {
        BigDecimal a = new BigDecimal("0");
        BigDecimal b = new BigDecimal("0");
        BigDecimal c = new BigDecimal("0");
         Scanner s = new Scanner(System.in);
         long n;
         n = s.nextInt();

         for (int i=0; i<n; i++)
         {
             a = s.nextBigDecimal();
             b                                                    =
a.multiply( a.subtract(BigDecimal.valueOf(1)));
             b  =  b.multiply(a.subtract(BigDecimal.valueOf(2)));
             b                                                    =
b.multiply(a.subtract(BigDecimal.valueOf(3)));
             b = b.divide(BigDecimal.valueOf(24));
             c  =  a.multiply(a.subtract(BigDecimal.valueOf(1)));
             c = c.divide(BigDecimal.valueOf(2));
             b = b.add(c);
             b = b.add(BigDecimal.valueOf(1));

             System.out.printf("%s\n", b.toString());
         }
    }
```

```
//Output :
```


Output

```
java -cp /tmp/msBZMyeG5F Main
4
1
1
2
2
3
4
4
8
```