

FUNCTIONS AND RECURSIONS

1. Problem Statement :

Some prime numbers can be expressed as a sum of other consecutive prime numbers. For example $5 = 2 + 3$, $17 = 2 + 3 + 5 + 7$, $41 = 2 + 3 + 5 + 7 + 11 + 13$. Now the task is to find out how many prime numbers which satisfy this property are present in the range 3 to N subject to a constraint that summation should always start with number 2.

Input Format:

First line contains a number N

Output Format:

Print the total number of all such prime numbers which are less than or equal to N.

Constraints:

$2 < N \leq 12,000,000,000$

Sample Input 1 :

20

Sample Output 1 :

2

Explanation :

$5 = 2 + 3$

$17 = 2 + 3 + 5 + 7$

Sample Input 2 :

15

Sample Output 2 :

1

Explanation :

$5 = 2 + 3$

Solution:

C:

```
#include <stdio.h>
int prime(int b)
{
    int j,cnt;
    cnt=1;
```

```
for(j=2;j<=b/2;j++)
{
    if(b%j==0)
        cnt=0;
}
if(cnt==0)
    return 1;
else
    return 0;
}
int main() {
    int i,j,n,cnt,a[25],c,sum=0,count=0,k=0;
    scanf("%d",&n);
    for(i=2;i<=n;i++)
    {
        cnt=1;
        for(j=2;j<=n/2;j++)
        {
            if(i%j==0)
                cnt=0;
        }
        if(cnt==1)
        {
            a[k]=i;
            k++;
        }
    }
    for(i=0;i<k;i++)
    {
        sum=sum+a[i];
        c= prime(sum);
        if(c==1)
            count++;
    }
    printf("%d",count);
    return 0;
}
```

```
}
```

Java:

```
import java.util.Scanner;
class Main {
    static int prime(int b) {
        int j,cnt;
        cnt=1;
        for (j = 2; j <= b/2; j++) {
            if(b%j==0)
                cnt=0;
        }
        if(cnt==0)
            return 1;
        else
            return 0;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int i,j,n=0,cnt,c=0,sum=0,count=0,k=0;
        Main t = new Main();
        int[] a = new int[25];
        System.out.println("Enter no");
        n = sc.nextInt();
        for (i = 2; i <=n ; i++) {
            cnt=1;
            for (j = 2; j <= n/2; j++) {
                if(i%j==0)
                    cnt=0;
            }
            if(cnt==1) {
                a[k]=i;
                k++;
            }
        }
        for (i = 0; i < k; i++) {
            sum=sum+a[i];
        }
    }
}
```

```

c=t.prime(sum);
if(c==1)
count++;
}
System.out.println(count);
}
}

```

Python:

```

num = int(input())
arr = []
sum = 0
count = 0
if num > 1:
    for i in range(2, num + 2):
        for j in range(2, i):
            if i % j == 0:
                break
        else:
            arr.append(i)
def is_prime(sum):
    for i in range(2, (sum // 2) + 2):
        if sum % i == 0:
            return False
    else:
        return True
for i in range(0, len(arr)):
    sum = sum + arr[i]
    if sum <= num:
        if is_prime(sum):
            count = count + 1
print(count)

```

2. Problem Statement :

A positive integer d is said to be a factor of another positive integer N if when N is divided by d , the remainder obtained is zero. For example, for number 12, there are 6 factors 1, 2, 3,

4, 6, 12. Every positive integer k has at least two factors, 1 and the number k itself. Given two positive integers N and k , write a program to print the k th largest factor of N .

Input Format:

The input is a comma-separated list of positive integer pairs (N, k)

Output Format:

The k th highest factor of N . If N does not have k factors, the output should be 1.

Constraints:

$1 < N < 10000000000$. $1 < k < 600$. You can assume that N will have no prime factors which are larger than 13.

Sample Input 1 :

12,3

Sample Output 1 :

4

Explanation :

N is 12, k is 3. The factors of 12 are (1,2,3,4,6,12). The highest factor is 12 and the third largest factor is 4. The output must be 4

Sample Input 2 :

30,9

Sample Output 2 :

1

Explanation :

N is 30, k is 9. The factors of 30 are (1,2,3,5,6,10,15,30). There are only 8 factors. As k is more than the number of factors, the output is 1.

Solution:

C:

```
#include<stdio.h>
#include<math.h>
int kPrimeFactor(int n, int k)
{
while (n%2 == 0)
{
```

```

k--;
n = n/2;
if (k == 0)
return 2;
}
for (int i = 3; i <= sqrt(n); i = i+2)
{
while (n%i == 0)
{
if (k == 1)
return i;
k--;
n = n/i;
}
}
if (n > 2 && k == 1)
return n;
return -1;
}
int main()
{
int n = 12, k = 3;
printf("%d",kPrimeFactor(n, k));
return 0;
}

```

Java:

```

import java.io.*;
import java.math.*;
class GFG{
static int kPrimeFactor(int n, int k)
{
while (n % 2 == 0)
{
k--;
n = n / 2;
if (k == 0)

```

```

return 2;
}
for (int i = 3; i <= Math.sqrt(n); i = i + 2)
{
    while (n % i == 0)
    {
        if (k == 1)
            return i;
        k--;
        n = n / i;
    }
}
if (n > 2 && k == 1)
    return n;
return -1;
}
public static void main(String args[])
{
    int n = 12, k = 3;
    System.out.println(kPrimeFactor(n, k));
}
}

```

Python:

```

import math
def kPrimeFactor(n,k) :
    while (n % 2 == 0) :
        k = k - 1
        n = n / 2
    if (k == 0) :
        return 2
    i = 3
    while i <= math.sqrt(n) :
        while (n % i == 0) :
            if (k == 1) :
                return i
            k = k - 1

```

```

n = n / i
i = i + 2
if (n > 2 and k == 1) :
    return n
return -1
n = 12
k = 3
print(kPrimeFactor(n, k))

```

3. Problem Statement :

The problem solvers have found a new castle for coding and named it as Nik's castle. These smart people were given a task to make the purchase of items at the castle easier by distributing various coins with different values. Rahul has come up with a solution that if we make coins category starting from \$1 till the maximum price of the item present on castle, then we can purchase any item easily. He added following example to prove his point. Let's suppose the maximum price of an item is 5\$ then we can make coins of {\$1, \$2, \$3, \$4, \$5} to purchase any item ranging from \$1 to \$5.

Now Tej, being a keen observer suggested that we could actually minimize the number of coins required and gave following distribution {\$1, \$2, \$3}. According to him, any item can be purchased one time ranging from \$1 to \$5. Everyone was impressed with both of them. Your task is to help Tej your dear friend to come up with the minimum number of denominations for any arbitrary max price in castle.

Input Format:

First line contains an integer T denoting the number of test cases. Next T lines contains an integer N denoting the maximum price of the item present on Nik's castle.

Output Format:

For each test case print a single line denoting the minimum number of denominations of coins required.

Constraints:

$1 \leq T \leq 100$

$1 \leq N \leq 5000$

Sample Input 1 :

2
10
5

Sample Output 1 :

4
3

Explanation :

According to Rahul {\$1, \$2, \$3,... \$10} must be distributed. But as per Tej only {\$1, \$2, \$3, \$4} coins are enough to purchase any item ranging from \$1 to \$10. Hence minimum is 4. Likewise denominations could also be {\$1, \$2, \$3, \$5}.

Hence answer is still 4.

Sample Input 2 :

3
1
5
7

Sample Output 2 :

1
3
3

Explanation :

According to Rahul {\$1, \$2, \$3, \$4, \$5} must be distributed. But as per Tej only {\$1, \$2, \$3} coins are enough to purchase any item ranging from \$1 to \$5. Hence minimum is 3. Likewise denominations could also be {\$1, \$2, \$4}. Hence answer is still 3.

Solution:

C:

```
#include<stdio.h>
int main(){
int n;
scanf("%d",&n)
int x=0;
while(pow(2,x)<=n){
```

```

x++;
}
printf("%d", (x));
}
}

```

Java:

```

import java.io.*;
public class Main
{
    public static void main(String[] args) throws IOException
    {
        BufferedReader br=new BufferedReader(new
        InputStreamReader(System.in));
        int n=Integer.parseInt(br.readLine());
        int x=0;
        while(Math.pow(2,x)<=n){
            x++;
        }
        System.out.println(x);
    }
}

```

Python:

```

cases=int(input())
for i in range(1,cases+1):
    value=int(input())
    coincount = 0
    while value>=1:
        value=value//2
        coincount=coincount+1
    print (coincount)

```

4. Problem Statement :

Find the minimum number of coins required to form any value between 1 to N, both inclusive. Cumulative value of coins should not exceed N. Coin denominations are 1 Rupee, 2 Rupee and 5 Rupee.

Let's understand the problem using the following example. Consider the value of N is 13, then the minimum number of coins required to formulate any value between 1 and 13, is 6. One 5 Rupee, three 2 Rupee and two 1 Rupee coins are required to realize any value between 1 and 13. Hence this is the answer.

However, if one takes two 5 Rupee coins, one 2 rupee coins and two 1 rupee coins, then to all values between 1 and 13 are achieved. But since the cumulative value of all coins equals 14, i.e., exceeds 13, this is not the answer.

Input Format:

A single integer value.

Output Format:

Four Space separated Integer Values

1st – Total Number of coins

2nd – number of 5 Rupee coins.

3rd – number of 2 Rupee coins.

4th – number of 1 Rupee coins.

Constraints:

$0 < n < 1000$

Sample Input 1 :

13

Sample Output 1 :

6 1 3 2

C:

```
#include<stdio.h>
int main(){
    int number,one,two,five;
    scanf("%d",&number);
    five = ((number-4)/5);
    if(((number-5*five) % 2) == 0)
        one=2;
    else
        one=1;
    two=(number-5*five-one)/2 ;
```

```
printf("%d,%d,%d,%d",one+two+five,five,two,one);
}
```

Java:

```
import java.util.*;
public class Main{
public static void main(String []args){
int number,one,two,five;
Scanner ip=new Scanner(System.in);
number=ip.nextInt();
five = ((number-4)/5);
if(((number-5*five) % 2) == 0)
one=2;
else
one=1;
two=(number-5*five-one)/2 ;
System.out.printf("%d,%d,%d,%d",one+two+five,five,two,one);
}
}
```

Python:

```
number = int(input())
five = int((number-4)/5)
if((number-5*five) % 2) == 0:
one=2
else:
one=1
two=(number-5*five-one)//2
print(one+two+five,five,two,one)
```

5. Problem Statement :

Ram wanted to take loan, there are two banks – Bank A and Bank B. Their interest rates vary. He has received offers from both banks in terms of the annual rate of interest, tenure, and variations of the rate of interest over the entire tenure. Now, he need to choose the offer which costs him with less interest.

Help him to doing the computation and making a wise choice. The loan repayment happens at a monthly frequency and Equated Monthly Installment (EMI) is calculated using the formula given below :

$$\text{EMI} = \frac{\text{loanAmount} * \text{monthlyInterestRate}}{(1 - 1 / (1 + \text{monthlyInterestRate})^{(\text{numberOfYears} * 12)})}$$

Input Format:

- First line: P principal (Loan Amount)
- Second line: T Total Tenure (in years).
- Third Line: N1 is the number of slabs of interest rates for a given period by Bank A. First slab starts from the first year and the second slab starts from the end of the first slab and so on.
- Next N1 line will contain the interest rate and their period.
- After N1 lines we will receive N2 viz. the number of slabs offered by the second bank.
- Next N2 lines are the number of slabs of interest rates for a given period by Bank B. The first slab starts from the first year and the second slab starts from the end of the first slab and so on.
- The period and rate will be delimited by single white space.

Output Format:

Print either Bank A or Bank B

Constraints:

$$1 \leq P \leq 1000000$$

$$1 \leq T \leq 50$$

$$1 \leq N1 \leq 30$$

$$1 \leq N2 \leq 30$$

Sample Input 1 :

10000

20

3

5 9.5

10 9.6

5 8.5

3

10 6.9

5 8.5

5 7.9

Sample Output 1 :

Bank B

Sample Input 2 :

500000

26

3

13 9.5

3 6.9

10 5.6

3

14 8.5

6 7.4

6 9.6

Sample Output 2 :

Bank A

Solution:

C:

```
#include <stdio.h>
```

```
#include <math.h>
```

```
int main()
```

```
{
```

```
double p,s,mi,sum,emi,bank[5],sq;
```

```
int y,n,k,i,yrs,l=0;
```

```
scanf("%lf",&p);
```

```
scanf("%d",&y);
```

```
for(k=0;k<2;k++)
```

```
{
```

```
scanf("%d",&n);
```

```
sum=0;
```

```
for(i=0;i<n;i++)
```

```
{
scanf(" %d",&yrs);
scanf(" %lf",&s);
mi=0;
sq=pow((1+s),yrs*12);
emi= (p*(s))/(1-1/sq);
sum= sum + emi;
}bank[l++]=sum;
}
if(bank[0]<bank[1])
printf(" Bank A ");
else
printf(" Bank B ");
return 0;
}
```

Java:

```
import java.util.Scanner;
public class Main{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        double p,s,mi,sum,emi,sq;
        int y,n,k,yrs,l=0;
        double[] bank = new double[5];
        System.out.println("Enter the principal amount");
        p = sc.nextDouble();
        System.out.println("Enter tenature year");
        y = sc.nextInt();
        for (k = 0; k < 2; k++) {
            System.out.println("Enter the no of slabs");
            n = sc.nextInt();
            sum=0;
            for (int i = 0; i < n; i++) {
                System.out.println("Enter the period :");
                yrs = sc.nextInt();
                System.out.println("Enter the interest :");
                s = sc.nextDouble();
```

```

mi=0;
sq=Math.pow((1+s), yrs*12);
emi=(p*(s))/(1-1/sq);
sum=sum+emi;
}
bank[l++]=sum;
}
if(bank[0]<bank[1])
System.out.println("Bank A");
else
System.out.println("Bank B");
}
}

```

Python:

```

bank = []
principal = int(input())
year = span style="color: #ffcc00;">int(input())
for i in range(0, 2): # 2 Banks
    installments = int(input())
    sum = 0
    for i in range(0, installments):
        time, roi = [float(i) for i in input().split()]
        square = pow((1+roi), time*12)
        emi = (principal*(roi)/(1-1/square))
        sum = sum + emi
    bank.append(sum)
if bank[0] < bank[1]:
    print("Bank A")
else:
    print("Bank B")

```

6. Problem Statement :

Identify the logic behind the series

6 28 66 120 190 276.....

The numbers in the series should be used to create a Pyramid. The base of the Pyramid will be the widest and will

start converging towards the top where there will only be one element. Each successive layer will have one number less than that on the layer below it. The width of the Pyramid is specified by an input parameter N. In other words there will be N numbers on the bottom layer of the pyramid.

The Pyramid construction rules are as follows

1. First number in the series should be at the top of the Pyramid
2. Last N number of the series should be on the bottommost layer of the Pyramid, with Nth number being the right-most number of this layer.
3. Numbers less than 5-digits must be padded with zeroes to maintain the sanctity of a Pyramid when printed. Have a look at the examples below to get a pictorial understanding of what this rule actually means.

Example

If input is 2, output will be

00006

00028 00066

If input is 3, output will be

00006

00028 00066

00120 00190 00276

Input Format:

First line of input will contain number N that corresponds to the width of the bottom-most layer of the Pyramid

Output Format:

The Pyramid constructed out of numbers in the series as per stated construction rules

Constraints:

$0 < N \leq 14$

Solution:

C:

```
#include<stdio.h>
```

```
int main()
```

```
{
int x=6,y=22,n,i,j,k=0;
scanf("%d",&n);
if(n>0 && n<=14)
{
for(i=1;i<=n;i++)
{
for(j=i;j<n;j++)
{
printf(" ");
}
for(k=1;k<=i;k++)
{
printf("%05d ",x);
x=x+y;
y=y+16;
}
printf("\n");
}
}
return 0;
}
```

Java:

```
import java.util.*;
public class Main{
public static void main(String[] args)
{
int x=6,y=22,n,i,j,k=0;
Scanner ip=new Scanner(System.in);
n=ip.nextInt();
if(n>0 && n<=14)
{
for(i=1;i<=n;i++)
{
for(j=i;j<n;j++)
{
```

```

System.out.printf(" ");
}
for(k=1;k<=i;k++)
{
System.out.printf("%05d ",x);
x=x+y;
y=y+16;
}
System.out.printf("\n");
}
}
}
}

```

Python:

```

n=int(input())
if(n>0 && n<=14):
    for i in range(1,n+1):
        for j in range(i,n):
            print(" ")
        for k in range(1,i+1):
            print(x)
        x=x+y
        y=y+16

```

7. Problem Statement :

There's a staircase with N steps, and you can climb 1 or 2 steps at a time. Given N, write a function that returns the number of unique ways you can climb the staircase. The order of the steps matters.

For example, if N is 4, then there are 5 unique ways:

```

1, 1, 1, 1
2, 1, 1
1, 2, 1
1, 1, 2
2, 2

```

What if, instead of being able to climb 1 or 2 steps at a time,

you could climb any number from a set of positive integers X? For example, if $X = \{1, 3, 5\}$, you could climb 1, 3, or 5 steps at a time.

Generalize your function to take in X.

Solution:

Python:

```
def staircase(n, X):
    if n < 0:
        return 0
    elif n == 0:
        return 1
    elif n in X:
        return 1 + sum(staircase(n - x, X) for x in X if x < n)
    else:
        return sum(staircase(n - x, X) for x in X if x < n)
```

```
X = [1,3,5]
n=4
print(staircase(n,X))
```

8. Problem statement

We are given n disks and a series of rods, we need to transfer all the disks to the final rod under the given constraints–

- ✓ We can move only one disk at a time.
- ✓ Only the uppermost disk from the rod can be moved.
- ✓ Any bigger disk cannot be placed on the smaller disk

Input Format :

Input number of disks

Output Format :

Show the complete action of disks moving from one rod to other rod to reach destination

Sample Input 1 :

3

Sample Output 1 :

Disk 1 moved from A to C

Disk 2 moved from A to B
 Disk 1 moved from C to B
 Disk 3 moved from A to C
 Disk 1 moved from B to A
 Disk 2 moved from B to C
 Disk 1 moved from A to C

Solution:**C:**

```
#include <stdio.h>

void towerOfHanoi(int n, char from_rod, char to_rod, char
aux_rod)
{
    if (n == 1)
    {
        printf("\n Move disk 1 from rod %c to rod %c",
from_rod, to_rod);
        return;
    }
    towerOfHanoi(n-1, from_rod, aux_rod, to_rod);
    printf("\n Move disk %d from rod %c to rod %c", n,
from_rod, to_rod);
    towerOfHanoi(n-1, aux_rod, to_rod, from_rod);
}

int main()
{
    int n = 4;
    towerOfHanoi(n, 'A', 'C', 'B');
    return 0;
}
```

Java:

```
class GFG
{
    static void towerOfHanoi(int n, char from_rod, char
to_rod, char aux_rod)
    {
        if (n == 1)
```

```

{
System.out.println("Move disk 1 from rod " +
from_rod + " to rod " + to_rod);
return;
}
towerOfHanoi(n-1, from_rod, aux_rod, to_rod);
System.out.println("Move disk " + n + " from rod " +
from_rod + " to rod " + to_rod);
towerOfHanoi(n-1, aux_rod, to_rod, from_rod);
}
public static void main(String args[])
{
int n = 4;
towerOfHanoi(n, 'A', 'C', 'B');
}
}

```

Python:

```

def TowerOfHanoi(n , from_rod, to_rod, aux_rod):
if n == 1:
print "Move disk 1 from rod",from_rod,"to
rod",to_rod
return
TowerOfHanoi(n-1, from_rod, aux_rod, to_rod)
print "Move disk",n,"from rod",from_rod,"to
rod",to_rod
TowerOfHanoi(n-1, aux_rod, to_rod, from_rod)
n = 4
TowerOfHanoi(n, 'A', 'C', 'B')

```

Mathematical Computation Problem

1. Consider the first three natural numbers 1, 2, 3.

These can be arranged in the following ways: 2, 3, 1 and 1, 3, 2.

In both of these arrangements, the numbers increase to a certain point and then decrease.

A sequence with this property is called a "mountain peak sequence".

Given an integer N, write a program to find the remainder of mountain peak arrangements that can be obtained by rearranging the numbers 1, 2, ..., N.

Input Format:

One line containing the integer N

Output Format:

An integer m, giving the remainder of the number of mountain peak arrangements that could be obtained from 1, 2, ..., N is divide by Mod

Constraints:

$\text{Mod} = 10^9 + 7$

$N \leq 10^9$

Example 1

Input

3

Output

2

Explanation

There are two such arrangements: 1, 3, 2 and 2, 3, 1

Example 2

Input

4

Output

6

Explanation

The six arrangements are (1, 2, 4, 3), (1,3,4,2), (1,4,3,2), (2,3,4,1), (2,4,3,1), (3,4,2,1)

Solution:**C++:**

```
#include<iostream>

#include<algorithm>

#include<vector>

using namespace std;

int min(int a,int b)

{

    return a>b?a:b ;

}

int bino(int n,int r,int p)

{

    vector<int>C(r+1,0);

    C[0]=1;

    for(int i=1;i<=n;i++ )

    {

        for(int j=min(i,r);j>0;j--)

            C[j]=(C[j] + C[j-1])%p;

    }

    return C[r];

}
```



```
int main()
{
    int N,m=0;

    int mod;

    mod= 1000000007;

    cin>>N;

    for(int i=1;i<N-1;i++)
        m+=bino(N-1,i,mod);

    cout<<(m%mod);

    return 0;
}
```

Java:

```
import java.math.BigInteger;

import java.util.Scanner;

public class MountainPeakSequence {

    public static void main(String[] args) {

        Scanner s = new Scanner(System.in);

        int N = s.nextInt();

        if (N>2) {

            BigInteger two = new BigInteger("2");

            BigInteger modulo = new BigInteger("1000000007");
```

```

        System.out.println(two.modPow(BigInteger.valueOf(N-1),
modulo).subtract(two.mod(modulo)).mod(modulo));

    }

    else

        System.out.println("O");

    s.close();

}

}

```

Python:

```

from itertools import permutations

```

```

N = int(input())

numbers = list(range(1,N+1))

count = 0

perm = list(permutations(numbers))

for i in range(len(perm)) :

    result = False

    for x in range(0, len(perm[i])-1):

        if perm[i][x+1] > perm[i][x]:

            result = True

    else:

```

```

        break

    if result:

        for index in range(x, len(perm[i])-1):

            if (perm[i][index+1] < perm[i][index]):

                result = True

            else:

                result = False

                break

    if result:

        count = count + 1

print(count)

```

2. Tom the cat is brushing up his Math skills. He has a bag containing N balls of different colors. Now Tom can randomly pick any even number of balls from the bag. Tom wants to find out the sum of all such combinations of balls that he can pull out from the bag. He can pull out at max K balls in one pick.

Input Format:

First line contains two space separated numbers N and K

Output Format:

The output is the sum of all the combinations of balls he can pull out modulo 10^9+7 i.e. (1000000007)

Constraints:

$0 \leq N$

$k \leq 10^{14}$

$N \geq k$

Example 1 :

Input :

4 4

Output :

8

Explanation :

We need ${}^4C_0 + {}^4C_2 + {}^4C_4 = 1 + 6 + 1 = 8$

Solution:

C:

```
#include<stdio.h>

unsigned long long int fact(unsigned long long int);

int main()
{
    unsigned long long int k,s=0,i,n;

    scanf("%llu %llu",&n,&k);

    if(n>0 && n>=k && k<=10000000000000000)
    {
        for(i=0;i<=k;i+=2)
        {
            s+=(fact(n)/(fact(i)*fact(n-i)));
        }

        s = s % 1000000007;

        printf("%llu",s);
    }

    return 0;
```

```
}  
  
unsigned long long int fact(unsigned long long int n)  
{  
    if(n==1)  
        return 1;  
    else if(n==0) return 1;  
    else  
        return(n*fact(n-1));  
}
```

Java:

```
import java.util.*;  
  
public class Main  
{  
    public static void main(String []args)  
    {  
        int k,s=0,i,n;  
  
        Scanner ip=new Scanner(System.in);  
  
        n=ip.nextInt();  
  
        k=ip.nextInt();  
  
        if(n>0 && n>=k && k<=100000000)  
        {
```

```

for(i=0;i<=k;i+=2)
{
s+=(fact(n)/(fact(i)*fact(n-i)));
}

s = s % 1000000007;

System.out.printf("%d",s);
}

}

public static int fact(int n)
{
if(n==1)
return 1;
else if(n==0) return 1;
else
return(n*fact(n-1));
}
}

```

Python:

```

def nCr(n, r):

    return (fact(n) / (fact(r)

```

```
* fact(n - r)))
```

```
def fact(n):
    res = 1
    for i in range(2, n+1):
        res = res * i
    return res
```

```
n = int(input())
k = int(input())
ans = 0
for i in range(0,k+1):
    if i%2 == 0:
        ans = ans + nCr(n,i)
print(ans%1000000007)
```

3. Sheldon Cooper, Leonard Hofstadter and Penny decide to go for drinks at Cheese cake factory. Sheldon proposes to make a game out of this. Sheldon proposes as follows,

- To decide the amount of beverage they plan to consume, say X.
- Then order for a random number of different drinks, say {A, B, C, D, E, F} of quantities {a, b, c, d, e, f} respectively.
- If quantity of any three drinks add up to X then we'll have it else we'll return the order.

E.g. If $a + d + f = X$ then True else False

You are given

1. Number of bottles N corresponding to different beverages and hence their sizes
 2. Next N lines, contain a positive integer corresponding to the size of the beverage
 3. Last line consists of an integer value, denoted by X above
- Your task is to help find out if there can be any combination of three beverage sizes that can sum up to the quantity they intend to consume. If such a combination is possible print True else False

Input Format:

1. First line contains number of bottles ordered denoted by N
2. Next N lines, contains a positive integer A_i , the size of the i^{th} bottle
3. Last line contains the quantity they intend to consume denoted by X in text above

Output Format:

True, if combination is possible

False, if combination is not possible

Constraints:

$N \geq 3$

$A_i > 0$

$1 \leq i \leq N$

$X > 0$

Example 1 :

Input :

6

1 4 45 6 10 8

22

Output :

True

Explanation :

The sum of 2nd, 5th and 6th beverage size is equal to 22. So the output will be True.

Solution:

C:

```
#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

int main()

{

    int i,n,q,l,k,j,*a=NULL;

    scanf("%d",&n);

    a=(int*)calloc(n,sizeof(int));

    for(i=0;i<n;i++)

    {

        scanf("%d",&a[i]);

    }

    scanf("%d",&q);

    for(i=0;i<n;i++)

    {

        if(a[i]>q)

        {

            a[i]=a[n-1];

            n--;

            i--;

        }

    }
```

```
}  
for(i=0;i<n;i++)  
{  
    k=a[i]; //selecting 1st element  
    for(j=i+1;k+a[j]<q&& j<n;j++)  
    {  
        k=k+a[j];    //2nd element decided  
        for(l=j+1;l<n;l++)  
        {  
            if(k+a[l]==q)    //checking for 3rd element  
            {  
                printf("True\n");  
                goto end;  
            }  
        }  
        k=k-a[j];  
    }  
    k=k-a[i];  
}  
printf("False");  
  
end:getch();  
  
exit(0);
```

```
return 0;
```

```
}
```

Java:

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

```
class Main{
```

```
    void find3Numbers(int A[], int arr_size, int sum)  {
```

```
        int l, r;
```

```
        for (int i = 0; i < arr_size - 2; i++) {
```

```
            for (int j = i + 1; j < arr_size - 1; j++) {
```

```
                for (int k = j + 1; k < arr_size; k++) {
```

```
                    if (A[i] + A[j] + A[k] == sum) {
```

```
                        System.out.print("False");
```

```
                    }
```

```
                }
```

```
            }
```

```
        }
```

```
        System.out.print("True");
```

```
    }
```

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

```
    {
```

```
        Scanner sc = new Scanner(System.in);
```

```
        Main triplet = new Main();
```

```

        int arr_size = sc.nextInt();

        int sum = sc.nextInt();

        int A[] = new int[arr_size];

        for(int i = 0; i < arr_size; i++)

            A[i] = sc.nextInt();

        triplet.find3Numbers(A, arr_size, sum);

    }
}

```

Python:

```

def solve(n,q,x):

    sor=sorted(q)

    for i in range(n-2):

        for j in range(1,n-1):

            for k in range(2,n):

                if sor[i]+sor[j]+sor[k]==x:

                    return True

    return False

n=int(input())

q=[int(input()) for _ in range(n)]

x=int(input())

```

```
print(solve(n,q,x))
```

4. By nature, an average Indian believes in saving money. Some reports suggest that an average Indian manages to save approximately 30+% of his salary. Dhaniram is one such hard working fellow. With a view of future expenses, Dhaniram resolves to save a certain amount in order to meet his cash flow demands in the future.

Consider the following example.

Dhaniram wants to buy a TV. He needs to pay Rs 2000/- per month for 12 installments to own the TV. If let's say he gets 4% interest per annum on his savings bank account, then Dhaniram will need to deposit a certain amount in the bank today, such that he is able to withdraw Rs 2000/- per month for the next 12 months without requiring any additional deposits throughout.

Your task is to find out how much Dhaniram should deposit today so that he gets assured cash flows for a fixed period in the future, given the rate of interest at which his money will grow during this period.

Input Format:

First line contains desired cash flow **M**

Second line contains period in months denoted by **T**

Third line contains rate per annum **R** expressed in percentage at which deposited amount will grow.

Output Format:

Print total amount of money to be deposited now rounded off to the nearest integer

Constraints:

$M > 0$

$T > 0$

$R \geq 0$

Calculation should be done upto 11-digit precision

Test Cases:

Input:

500

3

12

Output:

1470

Solution:**C:**

```
#include<stdio.h>
int main() {
    int t,x,i;
    float temp, r,m;
    scanf("%f%d%f",&m,&t,&r);
    temp=m;
    while(t--)
    {
        temp=(temp*1200)/(1200+r);
        temp+=m;
    }
    x=(int)temp;
    printf("the amount he should pay is %d",x-(int)m);
    return 0;
}
```

Java:

```
Import java.util.Scanner;
```

```
Class Deposit{
```

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

```
int t,x,i;

float temp, r,m;

Scanner sc = new Scanner(System.in);
System.out.printf("%f%d%f",m,t,r);
temp=m;
while(t--)
{
temp=(temp*1200)/(1200+r);
temp+=m;
}
x=(int)temp;
System.out.printf("the amount he should pay is %d",x-(int)m);
}}
```

Python:

```
m= int(input())

T= int(input())

r= int(input())

Temp = m

while T>0:

    Temp=(Temp*1200)/(1200+r)

    Temp = Temp + m

    T = T-1

print(int(Temp)-int(m))
```

5. Problem Statement :

In NASA, two researchers, Mathew and John, started their work on a new planet, but while practicing research they faced a mathematical difficulty. In order to save the time they divided their work. So scientist Mathew worked on a piece and invented a number computed with the following formula:

A Mathew number is computed as follows using the formula:

$$H(n) = n(2n-1)$$

And scientist John invented another number which is built by the following formula which is called John number.

$$T(n) = n(n+1)/2$$

Now Mathew and John are jumbled while combining their work. Now help them combine their research work by finding out number in a given range that satisfies both properties. Using the above formula, the first few Mathew-John numbers are:

1 6 15 28 ...

Input Format:

Input consists of 3 integers T1,T2,M separated by space . T1 and T2 are upper and lower limits of the range. The range is inclusive of both T1 and T2. Find Mth number in range [T1,T2] which is actually Mathew-John number.

T1 T2 M, where T1 is upper limit of the range, T2 is lower limit of the range and M ,where Mth element of the series is required

Output Format:

For valid input, print Mth number from formed sequence between T1 and T2(inclusive) or no number is present at this index.

For invalid input, print invalid input.

Constraints:

$$0 < T1 < T2 < 1000000$$

Test Cases:

Input

90 150 2

Output:

120

Solution:

C:

```
#include <stdio.h>

int main() {

    int t1,t2,m,i,j,c=0,a,b,th[100],k=0;

    scanf("%d %d %d",&t1,&t2,&m);

    if(t1>0&&t2>0&&m>0)

    {

        for(i=1;i<=t2/2;i++)

        {

            a=i*((2*i)-1);

            for(j=1;j<=t2/2;j++)

            {

                b=j*(j+1)/2;

                if((a==b)&&(a>=t1&&a<=t2))

                    th[k++]=a;

            }

        }

    }
```

```
if(k<m)

    printf("No number is present at this index");

else

    printf("%d",th[m-1]);

}

else

    printf("Invalid Input");

return 0;

}
```

Java:

```
public class JumbleWithNumbers {

    public static void main(String[] args) {

        int lower = 0;

        int upper = 0;

        int n = 0;

        Scanner sc = new Scanner(System.in);

        String[] s=sc.nextLine().trim().split(" ");

        try {

            boolean flag = false;

            try {

                lower = Integer.parseInt(s[0].trim());
```

```
} catch (Exception e) {  
    flag = true;  
}  
  
try {  
    upper = Integer.parseInt(s[1].trim());  
  
} catch (Exception e) {  
    flag = true;  
}  
  
try {  
    n = Integer.parseInt(s[2].trim());  
} catch (Exception e) {  
    flag = true;  
}  
  
if (flag) {  
    throw new Exception();  
}  
  
if (lower > upper) {  
    throw new Exception();  
}
```

```
int i = 1;

flag = false;

int count = 0;

while (true) {

    int nu = i * (2 * i - 1);


    if (nu < lower) {

        i++;

        continue;

    }

    if (getMathew(lower,upper,nu)) {

        count++;

        if (count == n) {

            System.out.println(nu);

            break;

        }

    }


}


if (nu > upper) {

    flag = true;

    break;

}
```

```
    }  
    i++;  
}  
  
if (flag) {  
    System.out.println("No number is present at this index");  
}  
  
} catch (Exception e) {  
    System.out.println("Invalid Input");  
}  
}  
  
static boolean getMathew(int lower,int upper,int num) {  
    int i = 1;  
    while (true) {  
        int nu = i * (i + 1);  
        nu /= 2;  
        if (nu < lower) {  
            i++;  
            continue;  
        }  
        if (nu == num) {  
            return true;  
        }  
    }  
}
```

```

        if (nu > upper) {
            break;
        }
        i++;
    }
    return false;
} }

```

Python:

```

t1, t2, m = [int(x) for x in input().split()]
k = 0
th = []
if t1 > 0 and t2 > 0 and m > 0:
    for i in range(1,int(t2/2)):
        a = i*((2*i)-1)
        for j in range(1,int(t2/2)):
            b=j*(j+1)/2;
            if a==b and a >= t1 and a <= t2:
                th.append(a);
                k = k + 1
if k < m:
    print("No number is present at this index")
else:

```

```
print(th[m-1]);
```

else:

```
print("Invalid Input");
```

6. Problem Statement :

A sequence is said to be progressive if it doesn't decrease at any point in time.

For example 1 1 2 2 is a progressive sequence but 1 2 1 is not a progressive sequence. Let S be the sequence and be represented by L spaced integers K_i , now your task is to find out the first longest progressive sequence present in the given sequence (S).

Input Format:

First line will contain T , the length of the sequence and next line will contain T spaced integers K_i (where $i = 0, 1, \dots, L$).

Line 1 T , where T is the length of the sequence

Line 2 K_i , where K_i is integer in sequence separated by space

Constraints:

$1 \leq T \leq 10^6$ (one million)

$1 \leq K_i \leq 10^9$ (one billion)

Output Format:

Line 1 longest progressive sequence present in the given sequence

Test Cases:

Input:

4

1 1 2 1

Output:

1 1 2

C:

```
#include <stdio.h>
```

```
#include <string.h>
```

```
#include <stdlib.h>
```

```
int main()
```

```
{
```

```
int previous_len=0,n,a[100], start=0, c[10], len=0;
```

```
scanf("%d",&n);
```

```
for(int l = 0; l<n;l++)
```

```
scanf("%d",&a[l]);
```

```
for (int i = 0; i < (sizeof(a)/sizeof(int)); ++i) {
```

```
    if(a[i+1] > a[i]) {
```

```
        len++;
```

```
        if (len > previous_len) {
```

```
            previous_len=len;
```

```
            start=i+1-len;
```

```
        }
```

```
    } else {
```



```

        previous_len=len;

        len=0;

    }

}

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

    c[i]=a[start+i];

    printf("%d ",c[i]);

}

return 0;

}

```

Java:

```

import java.util.ArrayList;

import java.util.Scanner;


public class LongestProgressiveSequence {


    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        int t = sc.nextInt();
    }
}

```

```
ArrayList<Integer> ls = new ArrayList<Integer>();  
  
Integer[] temp = new Integer[0];  
  
for (int i = 0; i < t; i++) {  
  
    int num = sc.nextInt();  
  
    if (!(ls.isEmpty())) {  
  
        if (num >= ls.get(ls.size()-1 )) {  
  
            ls.add(num);  
  
        } else {  
  
            if(ls.size()>temp.length)  
                temp=ls.toArray(temp);  
            ls.clear();  
            ls.add(num);  
        }  
  
    } else {  
  
        ls.add(num);  
  
    }  
  
}
```

```
if(ls.size()>temp.length)
```

```

{
    temp=ls.toArray(temp);
    ls.clear();
}

for (int i = 0; i < temp.length; i++) {
    if(i<temp.length-1)
        System.out.print(temp[i]+" ");
    else
        System.out.println(temp[i]);

}

}

}

```

Python:

```

class Solution(object):

    def lengthOfLIS(self, nums):

        tails =[0 for i in range(len(nums))]

        size = 0

        for x in nums:

            i=0

```

```

j=size
while i!=j:
    mid = i + (j-i)//2
    if tails[mid]< x:
        i= mid+1
    else:
        j = mid
        tails[i] = x
        size = max(i+1,size)
        #print(tails)

return size

ob1 = Solution()
num = int(input())
print(ob1.lengthOfLIS(num))

```

7. Problem Statement :

In a conference, attendees are invited for a dinner after the conference. The Coordinator, Sagar arranged round tables for dinner and want to have an impactful seating experience for the attendees. Before finalizing the seating arrangement, he want to analyze all possible arrangements. There are R round tables and N attendees. In case where N is an exact multiple

of R , the number of attendees must be exactly N/R . If N is not an exact multiple of R , then the distribution of attendees must be as equal as possible. Please refer Example section for better understanding.

For example, $R = 2$ and $N = 3$

All possible seating arrangements are

(1,2) & (3)

(1,3) & (2)

(2,3) & (1)

Attendees are numbered from 1 to N .

Constraints:

$0 < R \leq 10$ (Integer)

$0 < N \leq 20$ (Integer)

Input Format:

One line containing two space delimited integers R and N , where R denotes the number of round tables and N denotes the number of attendees

Output Format:

Single integer S denoting number of possible unique arrangements

Test case:

Sample Input :

2 5

Sample Output :

10

Explanation :

$R = 2, N = 5$

1. (1,2,3) & (4,5)
2. (1,2,4) & (3,5)
3. (1,2,5) & (3,4)
4. (1,3,4) & (2,5)
5. (1,3,5) & (2,4)
6. (1,4,5) & (2,3)
7. (2,3,4) & (1,5)
8. (2,3,5) & (1,4)

9. (2,4,5) & (1,3)
10. (3,4,5) & (1,2)

Arrangements like

(1,2,3) & (4,5)

(2,1,3) & (4,5)

(2,3,1) & (4,5), etc.

But, as it is a round table, all the above arrangements are same

Solution:

C:

```
#include<stdio.h>
```

```
int ncr(int n, int r)
```

```
{
```

```
    int ans = 1;
```

```
    for (int i = 1; i <= r; i += 1)
```

```
    {
```

```
        ans *= (n - r + i);
```

```
        ans /= i;
```

```
    }
```

```
    return ans;
```

```
}
```

```
int main()
```

```
{
```

```
int r,n,ini,extra,t1,t2,com;

scanf("%d%d",&r,&n);

ini = n/r;

extra = n%r;

t1 = ini + 1;

t2 = ini;

com = 1;


for(int i = 0; i < extra; i++)
{
    com = com*ncr(n,t1);
    n = n-t1;
}


for(int i = extra; i < r; i++)
{
    com = com*ncr(n,t2);
    n = n - t2;
}

printf("%d",com);

}
```

Java:

```
class Combination
{
    static int ncr(int n, int r)
    {
        int ans = 1;

        for (int i = 1; i <= r; i += 1)
        {
            ans *= (n - r + i);
            ans /= i;
        }

        return ans;
    }

    public static void main(String[] args)
    {

        int r,n,ini,extra,t1,t2,com;

        Scanner sc = new Scanner(System.in);

        r = sc.nextInt();

        n = sc.nextInt();

        ini = n/r;
```



```
extra = n%r;

t1 = ini + 1;

t2 = ini;

com = 1;


for(int i = 0; i < extra; i++)
{
    com = com*ncr(n,t1);
    n = n-t1;
}


for(int i = extra; i < r; i++)
{
    com = com*ncr(n,t2);
    n = n - t2;
}

System.out.println(com);

}
```

```
}
```

Python:

```
def fact(n):
```

```
    f=1
```

```
for i in range(1,n+1):
    f=f*i
return f

def ncr(n,r):
    if n==r or r==0:
        return 1
    if r==1:
        return n
    res=fact(n)//fact(r)
    res=res//fact(n-r)
    return res

r,n=list(map(int,input().split()))
ini=n//r
extra=n%r
t1=ini+1
t2=ini
com=1
for i in range(0,extra):
    com=com*ncr(n,t1)
    n=n-t1
for i in range(extra,r):
    com=com*ncr(n,t2)
```

$n = n - t^2$

print(com)

8. Problem statement

In a global Mathematics contest, the contestants are told to invent some special numbers which can be built by adding the squares of its digits. Doing this perpetually, the numbers will end up to 1 or 4. If a positive integer ends with 1, then it is called the Number of Game.

An example from above is:

$$13 = 1^2 + 3^2 = 1 + 9 = 10 \text{ (Step:1)}$$

$$10 = 1^2 + 0^2 = 1 + 0 = 1 \text{ (Step:2), iteration ends in Step 2 since number ends with 1}$$

Then in next round, the contestants are asked to combine their newly invented number, i.e. Number of Game with prime number. Now, being a smart programmer, write a program to help the contestants to find out the Nth combined number within any given range, where N can be any integer.

Input Format:

Input consists of 3 integers X, Y, N, one on each line. X and Y are upper and lower limits of the range. The range is inclusive of both X and Y. Find Nth number in range [X,Y].

Output Format:

Output will show the Nth element of the combined series lying in the range between X and Y.

Constraints:

$$X \leq Y$$

$$X > 0$$

$$N > 0$$

Example 1

Input

1
30
3

Output

19

Example 2

Input

12
33
5

Output

No number is present at this index

Example 3

Input

-5
@
4

Output

Invalid Input

Solution:

C:

```
#include<stdio.h>
```

```
#include<stdbool.h>
```

```
#include<stdlib.h>
```

```
int isPrime(int n)
```

```
{  
    if(n == 1) return 0;  
    if(n == 2) return 1;  
    for(int i = 2; i <= n/2; i++)  
        if(n % i == 0)  
            return 0;  
  
    return 1;  
}
```

```
int sumDigitSquare(int n)  
{  
    int digit;  
    int sq = 0;  
    while (n) {  
        digit = n % 10;  
        sq += digit * digit;  
        n = n / 10;  
    }  
    return sq;  
}
```

```
bool Game(int n)
```

```
{  
  
while (1) {  
    if (n == 1)  
        return true;  
  
    n = sumDigitSquare(n);  
  
    if (n == 4)  
        return false;  
}  
  
return false;  
}  
  
int main(void)  
{  
    int x,y,n,count = 0,i;  
    scanf("%d\n%d\n%d",&x,&y,&n);  
  
    for( i = x; i <= y; i++)  
    {
```

```

    if(isPrime(i) && Game(i))
    {
        count++;
        if(count == n)
        {
            printf("%d ",i);
            exit(0);
        }
    }
}

if(i > y)
printf("No NUmber is present at this Index");
else
printf("Invalid Input");

}

```

Java:

```

public class GameOfPrimes {

    public static void main(String[] args) {

```

```
Scanner sc=new Scanner(System.in);  
  
try{  
  
    int lower=0,upper=0,n=0;  
  
    boolean flag=false;  
  
    try {  
        lower=Integer.parseInt(sc.nextLine().trim());  
  
    } catch (Exception e) {  
        flag=true;  
    }  
  
    try {  
        upper =Integer.parseInt(sc.nextLine().trim());  
  
    } catch (Exception e) {  
        flag=true;  
    }  
  
    try {  
        n=Integer.parseInt(sc.nextLine().trim());  
    } catch (Exception e) {  
        flag=true;  
    }  
}
```



```
if(flag)
    throw new Exception();
if(lower>upper || lower<=0 || n<=0)
    throw new Exception();

int count=0;
int i=lower;
for (i = lower; i <=upper; i++) {
    if(isPrime(i) && getJohn(i)==1)
    {
        count++;
        // System.out.println(count+" "+i+" "+getJohn(i));
        if(count==n)
        {
            System.out.println(i);
            break;
        }
    }

}

}

if(i>upper)
{
```

```
        System.out.println("No number is present at this index");
    }
    }catch(Exception e)
    {
        System.out.println("Invalid Input");
    }
}
```

```
public static boolean isPrime(int num) {
    boolean flag = false;
    if (num < 2) {
        return flag;
    }
    if (num == 2) {
        return true;
    }
    if (num % 2 == 0) {
        return flag;
    }
    int i = 3;
    int rootn = (int) Math.sqrt(num);
```

```

for (; i <= rootn; i += 2) {
    if (num % i == 0) {
        break;
    }

}

if (i > rootn) {
    flag = true;
}

return flag;
}

static int getJohn(int y)
{
    if(y<=4)
    {
        return y;
    }

    if(y==10)
        return 1;

    String s=y+"";

    char[] c=s.trim().toCharArray();

```

```
int sum=0;

for (int i = 0; i < c.length; i++) {

    sum+=Math.pow(Integer.parseInt(c[i]+"".trim()),2);

}
```

```
sum= getJohn(sum);

return sum;
```

```
}

}
```

Python:

```
def sumDigitSquare(n) :

    sq = 0

    while (n) :

        digit = n % 10

        sq = sq + digit * digit

        n = n // 10

    return sq
```

```
def Game(n) :
    while (1) :
        if (n == 1) :
            return True

        n = sumDigitSquare(n)

        if (n == 4) :
            return False

    return False
```

```
def isPrime(num):
    if num > 1:
        for i in range(2,num):
            if (num % i) == 0:
                return False
        return True
    else:
        False
```

```
count =0
```

```
x,y,n = [int(x) for x in input().split()]
```

```
for i in range(x,y+1):
```

```
    if(isPrime(i) and Game(i)):
```

```
        count = count + 1
```

```
    if count == n:
```

```
        print(i)
```

```
        break
```

```
if i > y:
```

```
    print("No NUmber is present at this Index")
```

ARRAYS

1. PROBLEM STATEMENT

Juan Marquinho is a geologist and he needs to count rock samples in order to send it to a chemical laboratory. He has a problem: The laboratory only accepts rock samples by a range of its size in ppm (parts per million).

Juan Marquinho receives the rock samples one by one and he classifies the rock samples according to the range of the laboratory. This process is very hard because the number of rock samples may be in millions.

Juan Marquinho needs your help, your task is to develop a program to get the number of rocks in each of the ranges accepted by the laboratory.

Input Format:

An positive integer S (the number of rock samples) separated by a blank space, and a positive integer R (the number of ranges of the laboratory); A list of the sizes of S samples (in ppm), as positive integers separated by space R lines where the i th line containing two positive integers, space separated, indicating the minimum size and maximum size respectively of the i th range.

Output Format:

R lines where the i th line containing a single non-negative integer indicating the number of the samples which lie in the i th range.

Constraints:

$10 \leq S \leq 10000$ $1 \leq R \leq 1000000$ $1 \leq \text{size of each sample (in ppm)} \leq 1000$

Example 1

Input: 10 2

345 604 321 433 704 470 808 718 517 811

300 350

400 700

Output: 2 4

Example 2

Input: 20 3

921 107 270 631 926 543 589 520 595 93 873 424 759 537 458
614 725 842 575 195

1 100

50 600

1 1000

Output: 1 12 20

C:

```
#include<stdio.h>
```

```
int main()
```

```
{
```

```
    int a[100],b[100][100],n,i,r,j,k;
```

```
    scanf("%d %d",&n,&r);
```

```
    for(i=0;i<n;i++)
```

```
    {
```



```
scanf("%d",&a[i]);  
}  
for(i=0;i<r;i++)  
{  
    for(j=0;j<=1;j++)  
    {  
        scanf("%d",&b[i][j]);  
    }  
}  
k=0;  
for(i=0;i<r;i++)  
{  
    int count=0;  
    for(j=0;j<n;j++)  
    {  
        if(a[j]>=b[i][k]&&a[j]<=b[i][k+1])  
        {  
            count++;  
        }  
    }  
    printf("%d\n",count);  
}
```

```
return 0;
```

```
}
```

Java:

```
import java.util.Scanner;
```

```
class Counting
```

```
{
```

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

```
    {
```

```
        int arr[] = new int[50],range_arr[] = new int[50];
```

```
        int size,range,i,j,count=0;
```

```
        Scanner sc = new Scanner(System.in);
```

```
        size = sc.nextInt();
```

```
        range = sc.nextInt();
```

```
        for(i = 0 ; i < size;i++)
```

```
            arr[i] = sc.nextInt();
```

```
        range=range*2;
```

```
        for( i = 0; i < range ; i = i + 2)
```

```

    {
        range_arr[i] = sc.nextInt();
        range_arr[i+1] = sc.nextInt();
        for(j=0; j<size ; j++)
        {
            if((arr[j] >= range_arr[i]) && (arr[j] <= range_arr[i+1]))
                count++;
        }
        System.out.printf("%d ",count);
        count=0;
    }

}

```

Python:

```

samples, ranges =[int(i) for i in input().split()]

count = 0

final = []

arr = list(map(int, input().split()))

for i in range(0, ranges):

    range1, range2 = [int(i) for i in input().split()]

    for j in range(0, samples):

```

```

    if range1 <= arr[j] <= range2:
        count = count + 1
    final.append(count)
    count = 0
for i in range(0, len(final)):
    print(final[i], end=" ")

```

2. PROBLEM STATEMENT

There are n houses build in a line, each of which contains some value in it.

A thief is going to steal the maximal value of these houses, but he can't steal in two adjacent houses because the owner of the stolen houses will tell his two neighbours left and right side.

What is the maximum stolen value?

Sample Input:

```
val[] = {6, 7, 1, 3, 8, 2, 5}
```

Sample Output:

```
20
```

C:

```
#include<stdio.h>
```

```
int max(int a, int b)
```

```

{
    if(a>b)
        return a;

    return b;
}

int maxLoot(int hval[], int n)
{
    if (n == 0)
        return 0;

    int value1 = hval[0];

    if (n == 1)
        return value1;

    int value2 = max(hval[0], hval[1]);

    if (n == 2)
        return value2;

    int max_val;

    for (int i=2; i<n; i++)
    {

        max_val = max(hval[i]+value1, value2);

        value1 = value2;
    }
}

```

```

        value2 = max_val;
    }

    return max_val;
}

int main()
{
    int hval[100],n;
    scanf("%d",&n);
    for(int i = 0; i <n; i++)
        scanf("%d",&hval[i]);
    printf("%d",maxLoot(hval, n));
    return 0;
}

```

Java:

```

import java.util.*;

class Main
{
    static int maxLoot(int hval[], int n)
    {
        if (n == 0)
            return 0;

        int value1 = hval[0];
    }
}

```

```
        if (n == 1)
            return value1;

        int value2 = Math.max(hval[0], hval[1]);

        if (n == 2)
            return value2;

        int max_val = 0;
        for (int i=2; i<n; i++)
        {
            max_val = Math.max(hval[i]+value1, value2);
            value1 = value2;
            value2 = max_val;
        }

        return max_val;
    }

    public static void main (String[] args)
    {
        int i;

        Scanner sc = new Scanner(System.in);

        int n = sc.nextInt();

        int hval[] = new int[50];

        for(i=0;i<n;i++)

            hval[i] = sc.nextInt();
```

```

        System.out.println( maxLoot(hval, n));
    }
}

```

Python:

```

def maximize_loot(hval, n):
    if n == 0:
        return 0

    value1 = hval[0]

    if n == 1:
        return value1

    value2 = max(hval[0], hval[1])

    if n == 2:
        return value2

    max_val = None

    for i in range(2, n):
        max_val = max(hval[i]+value1, value2)
        value1 = value2
        value2 = max_val

    return max_val

```

```

hval = []

```



```
n = int(input())  
for i in range(0,n):  
    ele = int(input())  
    hval.append(ele)  
  
n = len(hval)  
  
print( maximize_loot(hval, n))
```

3. PROBLEM STATEMENT

Krishna loves candies a lot, so whenever he gets them, he stores them so that he can eat them later whenever he wants to.

He has recently received N boxes of candies each containing C_i candies where C_i represents the total number of candies in the i th box. Krishna wants to store them in a single box. The only constraint is that he can choose any two boxes and store their joint contents in an empty box only. Assume that there are infinite number of empty boxes available.

At a time he can pick up any two boxes for transferring and if both the boxes say contain X and Y number of candies respectively, then it takes him exactly $X+Y$ seconds of time. As he is too eager to collect all of them he has approached you to tell him the minimum time in which all the candies can be collected.

Input Format:

- The first line of input is the number of test case T
- Each test case is comprised of two inputs
- The first input of a test case is the number of boxes N

- The second input is N integers delimited by whitespace denoting the number of candies in each box

Output Format:

Print minimum time required, in seconds, for each of the test cases.
Print each output on a new line.

Constraints:

- $1 \leq T \leq 10$
- $1 \leq N \leq 10000$
- $1 \leq [\text{Candies in each box}] \leq 100009$

S. No	Input	Output
1	1 4 1 2 3 4	19
2	1 5 1 2 3 4 5	34

C:

```
#include <stdio.h>
```

```
int main() {
```

```
int n,i,k=0,sum=0,s1=0,t,temp=0,j;
```

```
long c[100009],s[100009];
```

```
scanf("%d",&t);  
for(int l=0;l<t;l++)  
{  
    scanf("%d",&n);  
    for(i=0;i<n;i++)  
        scanf("%ld",&c[i]);  
    for(i=0;i<n;i++)  
    {  
        for(j=i+1;j<n;j++)  
        {  
            if(c[i]>c[j])  
            {  
                temp=c[i];  
                c[i]=c[j];  
                c[j]=temp;  
            }  
        }  
    }  
    sum=0;  
    k=0;  
    for(i=0;i<n;i++)  
    {
```

```
sum=sum+c[i];  
s[k]=sum;  
k++;  
}  
s1=0;  
for(i=1;i<k;i++)  
s1=s1+s[i];  
printf("%d",s1);  
}  
return 0;  
}
```

JAVA:

```
import java.util.Scanner;  
  
class Candies  
{  
    public static void main(String[] args)  
    {  
        Scanner sc = new Scanner(System.in);  
  
        int n,i,k=0,t,j;  
  
        long c[] = new long[1000000],s1=0,sum=0,temp=0,s[] = new  
        long[1000000];  
  
        t = sc.nextInt();
```

```
for(int l=0;l<t;l++)
{
    n = sc.nextInt();
    for(i=0; i<n; i++)
        c[i] = sc.nextInt();
    for(i=0;i<n;i++)
    {
        for(j=i+1;j<n;j++)
        {
            if(c[i]>c[j])
            {
                temp=c[i];
                c[i]=c[j];
                c[j]=temp;
            }
        }
    }
    sum=0;
    k=0;
    for(i=0;i<n;i++)
    {
        sum=sum+c[i];
```

```
        s[k]=sum;

        k++;

    }

    s1=0;

    for(i=1;i<k;i++)

        s1=s1+s[i];

    System.out.printf("%d",s1);

}

}

}
```

PYTHON:

```
k=0

sum=0

s1=0

temp=0

c = []

s = []

t= int(input())

for i in range(0,t):

    n = int(input())

    for i in range(0,n):
```

```

ele = int(input())

c.append(ele)

for i in range(0,n):

    for j in range(i+1,n):

        if(c[i]>c[j]):

            c[i],c[j]=c[j],c[i]

sum=0

k = 0

for i in range(0,n):

    sum=sum+c[i]

    print(k)

    s.append(sum)

    k = k + 1

s1=0

for i in range(1,k):

    s1=s1+s[i];

print(s1)

```

4. PROBLEM STATEMENT

TCS is working on a new project called "TestVita". There are **N** modules in the project. Each module (**i**) has completion time denoted in number of hours (**Hi**) and may depend on other modules. If Module **x** depends on Module **y** then one needs to complete **y** before **x**.

As Project manager, you are asked to deliver the project as early as possible.

Provide an estimation of amount of time required to complete the project.

Input Format:

First line contains **T**, number of test cases.

For each test case:

First line contains **N**, number of modules.

Next **N** lines, each contain:

(i) Module ID

(Hi) Number of hours it takes to complete the module

(D) Set of module ids that **i** depends on - integers delimited by space.

Output Format:

Output the minimum number of hours required to deliver the project.

Constraints:

1. $1 \leq T \leq 10$
2. $0 < N < 1000$; number of modules
3. $0 < i \leq N$; module ID
4. $0 < H_i < 60$; number of hours it takes to complete the module **i**
5. $0 \leq |D| < N$; number of dependencies

6. $0 < D_k \leq N$; module ID of dependencies

Test Cases:

Input

```
1
5
1 5 0
2 6 1
3 3 2
4 2 3
5 1 3
```

Output

```
16
```

C:

```
#include <stdio.h>
```

```
int main() {
```

```
    int n,m[10],t[10],d[10],q,a,i,sum=0;
```

```
    scanf("%d",&a);
```

```
    for(q=1;q<=a;q++)
```

```
    {
```

```
        scanf("%d",&n);
```

```
        for(i=0;i<n;i++)
```

```
            scanf("%d %d %d",&m[i],&t[i],&d[i]);
```

```
            for(i=0;i<n-1;i++)
```

```
            {
```

```
        if(d[i]==d[i+1])
        {
            if(t[i]<t[i+1])
            t[i]=0;
            else
            t[i+1]=0;
        }
    }
    for(i=0;i<n;i++)
    {
        sum=sum+t[i];
    }
    printf("%d",sum);
}

return 0;

}
```

JAVA:

```
import java.util.Scanner;

class TestVita
{
    public static void main(String[] args){

        Scanner sc = new Scanner(System.in);
```

```
int n,q,a,i,sum=0;

int m[] = new int[10],t[] = new int[10],d[] = new int[10];

a = sc.nextInt();

for(q=1;q<=a;q++)

{

n = sc.nextInt();

for(i=0;i<n;i++)

{

    m[i] = sc.nextInt();

    t[i] = sc.nextInt();

    d[i] = sc.nextInt();

}

for(i=0;i<n-1;i++)

{

    if(d[i]==d[i+1])

    {

        if(t[i]<t[i+1])

            t[i]=0;

        else

            t[i+1]=0;

    }

}

}
```

```
        for(i=0;i<n;i++)
        {
            sum=sum+t[i];
        }

        System.out.printf("%d",sum);
    }
}
```

Python:

```
sum = 0

m = []

t = []

d = []

a = int(input())

for q in range(1,a+1):

    n = int(input())

    for i in range(1,n+1):

        ele1,ele2,ele3 = [int(x) for x in input().split()]

        m.append(ele1)

        t.append(ele2)

        d.append(ele3)

    for i in range(0,n-1):
```

```

if (d[i]==d[i+1]):
    if (t[i] < t[i+1]):
        t[i]=0
    else:
        t[i+1]=0
for i in range(0,n):
    sum = sum + t[i]
print(sum)

```

5. PROBLEM STATEMENT

The task is to find the minimum sum of Products of two arrays of the same size, given that k modifications are allowed on the first array. In each modification, one array element of the first array can either be increased or decreased by 2.

Note- the product sum is Summation ($A[i]*B[i]$) for all i from 1 to n where n is the size of both arrays

Input Format:

1. First line of the input contains n and k delimited by whitespace
2. Second line contains the Array A (modifiable array) with its values delimited by spaces
3. Third line contains the Array B (non-modifiable array) with its values delimited by spaces

Output Format:

1. Output the minimum sum of products of the two arrays

Constraints:

1. $1 \leq N \leq 10^5$
2. $0 \leq |A[i]|, |B[i]| \leq 10^5$
3. $0 \leq K \leq 10^9$

Test Cases :

S.No.	Input	Output
1	3 5 1 2 -3 -2 3 -5	-31
2	5 3 2 3 4 5 4 3 4 2 3 2	25

C:

```
#include<stdio.h>

#include<math.h>

#include<stdlib.h>

int minproduct(int a[], int b[], int n, int k)
{
    int diff = 0, res = 0;

    int temp;
```

```
for (int i = 0; i < n; i++) {  
  
    int pro = a[i] * b[i];  
  
    res = res + pro;  
  
    if (pro < 0 && b[i] < 0)  
        temp = (a[i] + 2 * k) * b[i];  
    else if (pro < 0 && a[i] < 0)  
        temp = (a[i] - 2 * k) * b[i];  
    else if (pro > 0 && a[i] < 0)  
        temp = (a[i] + 2 * k) * b[i];  
    else if (pro > 0 && a[i] > 0)  
        temp = (a[i] - 2 * k) * b[i];  
  
    int d = abs(pro - temp);  
  
    if (d > diff)  
        diff = d;  
  
}  
  
return res - diff;  
  
}
```



```
int main()
```

```
{  
  
    int n,k,a[100],b[100];  
  
    scanf("%d %d", &n, &k);  
  
    for(int i = 0; i < n; i++)  
        scanf("%d",&a[i]);  
    for(int i = 0; i < n; i++)  
        scanf("%d",&b[i]);  
  
    /*    cout << minproduct(a, b, n, k)  
        << endl; */  
    printf("%d",minproduct(a, b, n, k));  
  
    return 0;  
}
```

JAVA:

```
import java.util.Scanner;
```

```
import java.math.*;
```

```
class MinProduct {
```

```
    static int minproduct(int a[], int b[], int n, int k)
```



```
{  
  
    int diff = 0, res = 0;  
  
    int temp = 0;  
  
    for (int i = 0; i < n; i++) {  
        int pro = a[i] * b[i];  
  
        res = res + pro;  
  
        if (pro < 0 && b[i] < 0)  
            temp = (a[i] + 2 * k) * b[i];  
        else if (pro < 0 && a[i] < 0)  
            temp = (a[i] - 2 * k) * b[i];  
        else if (pro > 0 && a[i] < 0)  
            temp = (a[i] + 2 * k) * b[i];  
        else if (pro > 0 && a[i] > 0)  
            temp = (a[i] - 2 * k) * b[i];  
  
        int d = Math.abs(pro - temp);  
  
        if (d > diff)  
            diff = d;  
    }  
  
    return res - diff;  
}
```

```
public static void main(String[] args)
{
    Scanner sc = new Scanner(System.in);

    int n = sc.nextInt();

    int k = sc.nextInt();

    int a[] = new int[n], b[] = new int[n];

    for(int i = 0; i < n ; i++)
        a[i] = sc.nextInt();
    for(int i = 0; i < n ; i++)
        b[i] = sc.nextInt();

    System.out.println(minproduct(a, b, n, k));
}
```

PYTHON:

```
def minproduct(a,b,n,k):

    diff = 0

    res = 0

    for i in range(n):

        pro = a[i] * b[i]
```

```
res = res + pro

if (pro < 0 and b[i] < 0):
    temp = (a[i] + 2 * k) * b[i]
elif (pro < 0 and a[i] < 0):
    temp = (a[i] - 2 * k) * b[i]
elif (pro > 0 and a[i] < 0):
    temp = (a[i] + 2 * k) * b[i]
elif (pro > 0 and a[i] > 0):
    temp = (a[i] - 2 * k) * b[i]
```

```
d = abs(pro - temp)
```

```
if (d > diff):
```

```
    diff = d
```

```
return res - diff
```

```
n,k = [int(x) for x in input().split()]
```

```
a = []
```

```
b = []
```

```
for i in range(n):
```

```
    ele1 = int(input())
```

```
    a.append(ele1)
```

```
for i in range(n):
```

```

ele2 = int(input())

b.append(ele2)

print(minproduct(a, b, n, k))

```

6. PROBLEM STATEMENT

Problem Description

There are N bottles. i th bottle has $A[i]$ radius. Once a bottle is enclosed inside another bottle, it ceases to be visible. Minimize the number of visible bottles.

You can put i th bottle into j th bottle if following condition is fulfilled:

1. i th bottle itself is not enclosed in another bottle.
2. j th bottle does not enclose any other bottle.
3. Radius of bottle i is smaller than bottle j (i.e. $A[i] < A[j]$).

Constraints :

- $1 \leq N \leq 100000$.
- $1 \leq A[i] \leq 10^{18}$.

Input Format :

- First line contains a single integer N denoting the number of bottles.
- Second line contains N space separated integers, i th integer denoting the radius of i th bottle. ($1 \leq i \leq N$).

Output Format:

Minimum number of visible bottles.

Test Cases :

Example 1

Input :

8

1 1 2 3 4 5 5 4

Output :

2

C:

```
#include <stdio.h>
```

```
int max(int a,int b)
```

```
{
```

```
    if(a>b)
```

```
        return a;
```

```
    return b;
```

```
}
```

```
int minBottles(int* arr, int n)
```

```
{
```

```
    int m[100] = {0};
```

```
    int ans = 0;
```

```
    for (int i = 0; i < n; i++)
```

```

    {
        m[arr[i]]++;
        ans = max(ans, m[arr[i]]);
    }
    return ans;
}

```

```

int main()
{
    int n,arr[100];
    scanf("%d", &n);

    for(int i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    printf("%d ",minBottles(arr, n));
    return 0;
}

```

JAVA:

```

import java.util.*;

import java.util.Scanner;

```

```
class MinBottles
{
    static void minBottles(int[] arr, int n)
    {
        HashMap<Integer, Integer> mp = new HashMap<Integer,
Integer>();
        int ans = 0;
        for (int i = 0; i < n; i++)
        {
            if (mp.containsKey(arr[i]))
                mp.put(arr[i], mp.get(arr[i]) + 1);
            else
                mp.put(arr[i], 1);

            ans = Math.max(ans, mp.get(arr[i]));
        }

        System.out.print(ans);
    }

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

```
Scanner sc = new Scanner(System.in);

int n = sc.nextInt();

int arr[] = new int[100];

for(int i = 0; i < n; i++)
    arr[i] = sc.nextInt();

    minBottles(arr, n);
}
}
```

PYTHON:

```
def minBottles(arr, n):

    m = dict()

    ans = 0

    for i in range(n):

        m[arr[i]] = m.get(arr[i], 0) + 1

        ans = max(ans, m[arr[i]])

    print(ans)

n = int(input())

arr = []
```



```
for i in range(n):  
    ele = int(input())  
    arr.append(ele)
```

```
minBottles(arr, n)
```

7. PROBLEM STATEMENT

Zoya has developed a new game called Zombie World. The objective of the game is to kill all zombies in given amount of time. More formally,

- **N** represents the total number of zombies in the current level
- **T** represents the maximum time allowed for the current level
- **P** represents the initial energy level a player starts with
- **E_i** defines the energy of the i-th zombie
- **D** defines the minimum energy the player needs, to advance to the next level

When a player energy is greater than or equal to the i-th zombie's energy, the player wins. Upon winning, the player will be awarded with an additional energy equal to the difference between current zombie energy and the player energy.

One unit of time will be taken to complete the fight with a single zombie.

Rules of the game:-

- At any given time, a player can fight with only one zombie
- Player is allowed to choose any one zombie to fight with.

Your task is to determine whether the player will advance to the next level or not, if he plays optimally.

Input Format:

The first line contains the number of test cases (K)

Each test case consists of three parts:

1. The total number of zombies (N) and the maximum time allowed (T)
2. Array of size N, which represents the energy of zombies (E)
3. The initial energy level a player (P) and the minimum energy required to advance (D)

Output Format:

Print "Yes" if a player can advance to the next level else print "No".

Constraints:

$$1 \leq K \leq 10$$

$$1 \leq N \leq 50$$

$$1 \leq E_i \leq 500$$

$$1 \leq T \leq 100$$

$$1 \leq D \leq 2000$$

$$1 \leq P \leq 500$$

Test Cases :

Input

1

2 3

4 5

5 7

Output

Yes

C:

```
#include<stdio.h>

int main()

{

    int n,t,e[20],i,pe,me,k;

    scanf("%d",&k);

    while(k)

    {

        scanf("%d",&n);

        scanf("%d",&t);

        for(i=0;i<n;i++)

            scanf("%d",&e[i]);

        scanf("%d",&pe);

        scanf("%d",&me);

        if(t<n)

            goto x;

        else

        {

            for(i=0;i<n;i++)

            {
```

```
    if(pe>=e[i])
    {
        pe=pe+(pe-e[i]);
    }
}

if(pe<=me)
printf("yes\n");

else
x: printf("no\n");

}

    k--;

}

return 0;

}
```

JAVA:

```
import java.util.Scanner;

class Zombie{

public static void main(String[] args)

{

    Scanner sc = new Scanner(System.in);

    int n,t,e[] = new int[20],i,pe,me,k;
```

```
k = sc.nextInt();  
while(k>0)  
{  
    n = sc.nextInt();  
    t = sc.nextInt();  
    for(i=0;i<n;i++)  
        e[i] = sc.nextInt();  
  
    pe = sc.nextInt();  
    me = sc.nextInt();  
    if(t<n){  
        System.out.printf("no\n");  
        k--;  
        continue;  
    }  
    else  
    {  
        for(i=0;i<n;i++)  
        {  
            if(pe>=e[i])  
            {  
                pe=pe+(pe-e[i]);  
            }  
        }  
    }  
}
```

```
}  
  
}  
  
if(pe<=me)  
    System.out.printf("yes");  
  
else  
    System.out.printf("no\n");  
  
}  
  
    k--;  
  
}  
  
  
}  
  
}
```

PYTHON:

```
e = []  
  
k = int(input())  
  
while k>0:  
    n, t = [int(x) for x in input().split()]  
  
    for i in range(0,n):  
        ele = int(input())  
        e.append(ele)  
  
    pe, me = [int(x) for x in input().split()]  
  
    if (t<n):
```

```

print("No")

k = k-1

continue

else:

    for i in range(0,n):

        if (pe>=e[i]):

            pe=pe+(pe-e[i])

        if (pe<=me):

            print("yes");

        else:

            printf("no\n");

k = k-1

```

8. PROBLEM STATEMENT

Race is generally organized by distance but this race will be organized by time. In order to predict the winner we will check every 2 seconds.

Let's say total race time is 7 seconds we will check for (7-1) seconds. For 7 sec : we will check who is leading at 2 sec, 4 sec, and 6 sec. Participant who is leading more number of times is winner from prediction perspective.

Now our task is to predict a winner in this marathon.

Note :

1. At particular time let say at 4th second, top two (top N, in general) participants are at same distance, then in this case both are leading we'll increase count for both(all N).
2. And after calculating at all time slices, if the number of times someone is leading, is same for two or more participants, then one who come first in input sequence will be the winner.

Ex. If participant 2 and 3 are both leading with same number, participant 2 will be the winner.

Constraints :

$$1 \leq T \leq 100$$

$$1 \leq N \leq 100$$

Input Format :

First line contains a single integer N denoting the number of participants.

Second line contains a single integer T denoting the total time in seconds of this Marathon.

Next N lines (for each participant) are as follows :

We have T+1 integers separated by space.

First T integers are as follows :

i-th integer denotes number of steps taken by the participant at the i-th second.

T+1st integer denotes the distance (in meters) of each step.

Output Format :

Index of Marathon winner, where index starts with 1.

Test Case :

Input

3
8
2 2 4 3 5 2 6 2 3
3 5 7 4 3 9 3 2 2
1 2 4 2 7 5 3 2 4

Output

2

C++:

```
#include
```

```
#include
```

```
#include
```

```
using namespace std;
```

```
int max_of_sublist(int sum_list[])
```

```
{
```

```
int k=INT_MIN;
```

```
int z=sizeof(sum_list)/sizeof(sum_list[0]);
```

```
for(int i=0;i<z;i++)
```

```
{
```

```
if(k<sum_list[i])
```

```
k=sum_list[i];
```

```
}
```

```
return k;
```

```

}

int who_win(int winer[])
{

int k;

int m=winer[0];

int z=sizeof(winer)/sizeof(winer[0]);

for(int i=0;i<z;i++)
{
if(m>t;

int n;

cin>>n;

int step[t][n];

for(int i=0;i<t;i++)

for(int j=0;j>step[i][j];

int winer[t];

int sum_list[t]={0};

for(int second=2;second<=n;second+2)

{

for(int pn=0;pn<t;pn++)

{

```

```
sum_list[pn]+=(step[pn][second-2]+step[pn][second-1])*step[pn][n];
```

```
}
```

```
int z;
```

```
z=max_of_sublist(sum_list);
```

```
for(int x=0;x<t;x++)
```

```
{
```

```
if(sum_list[x]==z)
```

```
winer[x]+=1;
```

```
}
```

```
}
```

```
cout<<who_win(winer)+1;
```

```
return 0;
```

```
}
```

Java:

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

```
import java.util.Arrays;
```

```
class Main
```

```
{
```

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

    BufferedReader br = new BufferedReader (new
InputStreamReader (System.in));

    int participants = Integer.parseInt (br.readLine ());

    int seconds = Integer.parseInt (br.readLine ());

    int record[][] = new int[participants][seconds + 1];


    int lead[] = new int[participants];

    Arrays.fill (lead, 0);

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

        String input = br.readLine ();

        input.trim ();


        input = input.replaceAll ("\\s+", "");


        if (input.length () != seconds + 1)

            System.exit (0);
```

```
for (int j = 0; j < input.length (); j++)  
{  
    record[i][j] = Character.getNumericValue (input.charAt (j));  
}  
}
```

```
for (int i = 0; i < participants; i++)  
{  
    int length = record[i][seconds];  
    for (int j = 0; j < seconds - 1; j++)  
    {  
        if (j != 0)  
        {  
            record[i][j] = record[i][j - 1] + (record[i][j] * length);  
        }  
        else  
        {  
            record[i][j] = (record[i][j] * length);  
        }  
    }  
}
```

```
}
```

```
int start = (seconds % 2 == 0) ? 0 : 1;
```

```
int max = 0;
```

```
for (int i = start; i <= seconds - 2; i += 2)
```

```
{
```

```
    for (int j = 0; j < max;)
```

```
        max = record[j][i];
```

```
    for (int k = 0; k < participants; k++)
```

```
        if (record[k][i] == max)
```

```
            lead[k]++;
```

```
        max = 0;
```

```
}
```

```
max = 0;
```

```
for (int i = 0; i < max;)
```

```
    max = lead[i];
```

```
for (int i = 0; i < participants; i++)
```

```
{
```

```

        if (lead[i] == max)
        {
            System.out.println (i + 1);

            break;
        }
    }
}

```

Python:

```

n = int (input ())
t = int (input ())
ply = []
for i
in range (n):
x = list (map (int, input ().split ()))
y = x[-1]
x = x[:-1]
    x[0] = x[0] * y
    for i
in range (1, t):
x[i] *= y
x[i] += x[i - 1]
ply.append (x)
if t
    %2 == 0:
t -= 2
    else :

```

```
t -= 1
```

```
d =
```

```
{
```

```
}
```

```
for i in range (1, n + 1):
```

```
d[i] = 0
```

```
for i
```

```
in range (2, t + 1, 2):
```

```
m = 0
```

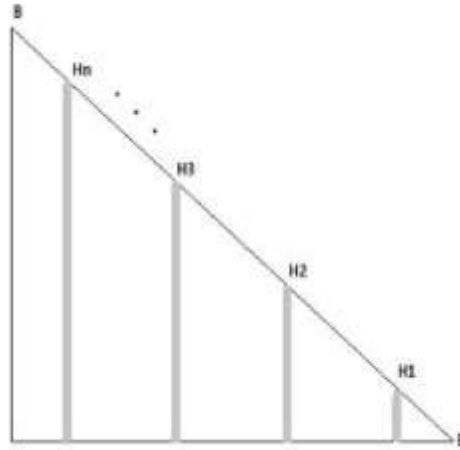
```
t1 = []
```

```
for j in range (n):
```

```
x = ply[j][i] b
```

ADVANCED ARRAYS

1. A man is doing an experiment with the device that he built newly. The structure of the device is as below diagram.



B to E is a sloping surface with n holes, labelled H_1, H_2, \dots, H_n , on it. Holes are of different diameters and depths. The man is releasing m number of balls of different diameters from the point B one after the other. He needs to find the positions of each ball after the experiment.

The specialities of the device are :

A ball will fall into the hole, if its diameter is less than or equal to the diameter of the hole.

A hole H_i will become full, if i numbers of balls fall into it. For example hole labelled H_3 will become full if 3 balls fall into it.

If a hole is full, then no more balls fall into it.

A ball will reach the bottom point E from B, if and only if it is not falling into any of the holes.

Please help him in finding the eventual position of the balls. If a ball is in hole P_i , then take its position as i . If a ball reached the bottom point E, then take its position as 0.

Constraints

$$0 \leq N \leq 50$$

$$0 < \text{Diameter of holes} \leq 10^9$$

$$0 < m \leq 1000$$

$$0 < \text{Diameter of balls} \leq 10^9$$

Input

Line 1 : total number of holes, N

Line 2 : N space separated integers denoting the diameters of N holes, from bottom to top.

Line 3 : total number of balls, M.

Line 4 : M space separated integers denoting diameters of balls in the order of release.

Output

Line 1 : Positions of each ball in the order of ball release separated by space.

Testcase**Input**

```
3
21 3 6
11
20 15 5 7 10 4 2 1 3 6 8
```

Output

```
1 0 3 0 0 3 3 2 2 0 0
```

Solution:**C:**

```
#include<stdio.h>
```

```
#define int long long
```

```
signed main(){
```

```
    int n;
```

```
    scanf("%lld",&n);
```

```
    int cap[n+1];
```

```
    int arr[n+1];
```

```
    arr[0]=1000000007;
```

```

cap[0]=100;

for(int i=1;i<=n;i++) {
    scanf("%lld",&arr[i]) ;

    cap[i]=i;
}

int q;

scanf("%lld",&q);

while(q--){

    int k;

    scanf("%lld",&k);

    for(int j=n;j>=0;j--){

        if(arr[j]>=k&&cap[j]>0){

            cap[j]--;

            printf("%lld",j);

        }

    }

}

```

```

}

```

Java:

```

import java.util.*;

public class Main{

```

```
public static void main(String[] args){  
    Scanner ip=new Scanner(System.in);  
  
    int n,i,q;  
  
    n=ip.nextInt();  
  
    int[] cap=new int[n+1];  
  
    int[] a=new int[n+1];  
  
    a[0]=1000000007;  
  
    cap[0]=100;  
  
    for( i=1;i<=n;i++) {  
        a[i]=ip.nextInt();  
  
        cap[i]=i;  
    }  
  
    q=ip.nextInt();  
  
    while(q!=0){  
        int k;  
  
        k=ip.nextInt();  
  
        for(int j=n;j>=0;j--){  
            if(a[j]>=k&&cap[j]>0){  
                cap[j]--;  
  
                System.out.print(j);  
  
            }  
  
        }q--;  
    }
```

```

    }

}
}

```

Python:

```

N=int(input())

if(0<N<=50):

    holes=[]*N

    holes=list(map(int,input().split()))

    H=[]

    l1=[]

    l2=[]

    final_list=[]

    for i in range(N):

        H.append(i+1)

    M=int(input())

    set_flag=0

    l1=list(range(0,N))

    l2=l1[::-1]

    if(0<M<=1000):

        balls=[]*M

        balls=list(map(int,input().split()))

        flag=1

```

```

for each_ball in balls:
    if(flag==0):
        for i in l2:
            if(each_ball<=holes[i] and H[i]>0):
                final_list.append(i+1)
                H[i]=H[i]-1
                set_flag=1
                flag=0
                break
            else:
                set_flag=0
        if(set_flag==0):
            final_list.append(0)
            flag=0
    if(flag==1):
        for j in l1:
            if(each_ball<=holes[j] and H[j]>0):
                final_list.append(j+1)
                H[j]=H[j]-1
                set_flag=1
                flag=1
                break

```

```

else:
    set_flag=0
if(set_flag==0):
    final_list.append(0)
    flag=0
for ii in range(M-1):
    print(final_list[ii],end=" ")
print(final_list[M-1])

```

2. You have been given 'N' number of spherical Balloons of different radius when filled. You have to fill one balloon per day and the last balloon will be filled on Nth day. There is some rate of air reduction 'K' per day from each balloon. Fill the balloons in such an order so that the sum of the volume of all the balloons is maximum on the day when all the balloons are filled.

Input Format: First Line is an integer N giving the number of balloons. Second line gives space separated N positive real numbers with up to 1 decimal place giving the radii of the balloons. Third line gives K, the rate of reduction in the volume of air as a percentage.

Output Format: Maximum sum of volumes of all the balloons on the Nth day when all the balloons are filled. Take 3.14 as the value of π and give the answer to two decimal places (truncated by ignoring all the decimals from third onwards). Note that the truncation should

happen only after computing the volume of all the balloons on the final day to maximum precision.

Constraints:

Number of balloons ≤ 10 Radius of balloons ≤ 200

Example 1

Input

5
8 4 6 10 3 1 0

Output

7117.88

Explanation

If we fill the balloons in the order 3, 4, 6, 8, 10, their volumes on the fifth day are respectively 7 4.165544 1 95.33312 7 32.4992 1929.216 4 186.666667. And their sum is 7117.880531. Truncating the value two decimal places, we obtain 7117.88

Example 2

Input

7
3 .5 9 4 6.6 7 11 9.1 1 2.5

Output

12555.35

Explanation

If we inflate the balloons in the order 3.5 4 6.6 7 9 9.1 11, their volumes on the seventh day would be respectively 8 0.56025567 1 37.4322396 705.5574848 9 62.0256771 2 336.74875 2 760.581763 5572.453333. The sum of these volumes is 12555.3595 and truncating to two decimal places, we obtain 12555.35

Solution:

C:


```
#include <stdio.h>

double rate(double y,float m,int k)
{
    for(int i=k;i>=1;i--)
    {
        y=y-y*(m/100);
    }
    return y;
}

int main() {
    int n,i,j;

    double v,b[1000],t,v1,sum=0;

    float a[1000],x;

    scanf("%d",&n);

    for(i=0;i<n;i++)

        scanf("%f",&a[i]);

        scanf("%f",&x);

        for(i=0;i<n;i++)

            for(j=i+1;j<n;j++)

                if(a[i]>a[j])

                {

                    t=a[i];
```

```

    a[i]=a[j];

    a[j]=t;

}

for(i=0;i<n;i++)

{

v=0;

v=(4*3.14*a[i]*a[i]*a[i])/3;

b[i]=rate(v,x,n-(i+1));

}

for(i=0;i<n;i++)

sum=sum+b[i];

printf("%.2f ",sum);

return 0;

}

```

Java:

```

import java.util.*;

public class Main{

public static double rate(double y,float m,int k)

{

    for(int i=k;i>=1;i--)

    {

```

```
y=y-y*(m/100);  
}  
return y;  
}  
  
public static void main(String[] args) {  
  
    int n,i,j;  
  
    double v,t,v1,sum=0;  
  
    double[] b=new double[1000];  
  
    float[] a=new float[1000];  
  
    float x;  
  
    Scanner ip=new Scanner(System.in);  
  
    n=ip.nextInt();  
  
    for(i=0;i<n;i++)  
  
        a[i]=ip.nextFloat();  
  
    x=ip.nextFloat();  
  
    for(i=0;i<n;i++)  
  
        for(j=i+1;j<n;j++)  
  
            if(a[i]>a[j])  
  
            {  
  
                t=a[i];  
  
                a[i]=a[j];  
  
                a[j]=(float)t;  
            }  
        }  
    }  
}
```

```
}  
  
for(i=0;i<n;i++)  
{  
    v=0;  
  
    v=(4*3.14*a[i]*a[i]*a[i])/3;  
  
    b[i]=rate(v,x,n-(i+1));  
}  
  
for(i=0;i<n;i++)  
  
    sum=sum+b[i];  
  
System.out.printf("%.2f ",sum);  
  
}  
  
}
```

Python:

```
def rate(y,m,k):  
    for i in range(k):  
        y=y-y*(m//100)  
  
    return y  
  
b=[]  
  
n=int(input())  
  
a=list(map(int,input().split()))  
  
x=int(input())
```

```

a.sort()

for i in range(0,n):

    v=0

    v=(4*3.14*a[i]*a[i]*a[i])/3;

    b.append(rate(v,x,n-(i+1)))

print(sum(b))

```

3. N monkeys are invited to a party where they start dancing. They dance in a circular formation, very similar to a Gujarati Garba or a Drum Circle. The dance requires the monkeys to constantly change positions after every 1 second.

The change of position is not random & you, in the audience, observe a pattern. Monkeys are very disciplined & follow a specific pattern while dancing.

Consider $N = 6$, and an array `monkeys = {3,6,5,4,1,2}`.

This array (1-indexed) is the dancing pattern. The value at `monkeys[i]`, indicates the new of position of the monkey who is standing at the *i*th position.

Given N & the array `monkeys[]`, find the time after which all monkeys are in the initial positions for the 1st time.

Constraints

$1 \leq t \leq 10$ (test cases)

$1 \leq N \leq 10000$ (Number of monkeys)

Input Format

First line contains single integer t , denoting the number of test cases.

Each test case is as follows -

Integer N denoting the number of monkeys.

Next line contains N integer denoting the dancing pattern array, `monkeys[]`.

Output

t lines,

Each line must contain a single integer T , where T is the minimum number of seconds after which all the monkeys are in their

initial position.

Test Case

Example 1

Input

1
6
3 6 5 4 1 2

Output

6

Explanation

Consider $N = 6$, and an array monkeys = {3,6,5,4,1,2}.

Suppose monkeys are a,b,c,d,e,f, & Initial position (at $t = 0$) ->

a,b,c,d,e,f

At $t = 1$ -> e,f,a,d,c,b

a will move to 3rd position, b will move to 6th position, c will move to 5th position, d will move to 4th position, e will move to 1st position and f will move to 2nd position. Thus from a,b,c,d,e,f at $t = 0$, we get e,f,a,d,c,b at $t = 1$. Recursively applying same transpositions, we get following positions for different values of t .

At $t = 2$ -> c,b,e,d,a,f

At $t = 3$ -> a,f,c,d,e,b

At $t = 4$ -> e,b,a,d,c,f

At $t = 5$ -> c,f,e,d,a,b

At $t = 6$ -> a,b,c,d,e,f

Since at $t = 6$, we got the original position, therefore the answer is 6.

Solution:

C:

```
#include <stdio.h>
```

```
int isequal(int x[],int m)
{
    int c1=0;
    for(int k=1;k<=m;k++)
    {
        if(x[k]==k)
            c1++;
    }
    return c1;
}

int main() {
    int n,a[100],i,b[100],k,c[100],cnt,t;

    scanf("%d",&t);

    for(k=0;k<t;k++)
    {
        cnt=0;

        scanf("%d",&n);

        for(i=1;i<=n;i++)
        {
            scanf("%d",&a[i]);

            b[i]=i;
```

```
}  
z: for(i=1;i<=n;i++)  
    c[a[i]]=b[i];  
cnt++;  
if(isequal(c,n)==n)  
    printf("%d\n",cnt);  
else  
{  
    for(i=1;i<=n;i++)  
        b[i]=c[i];  
    goto z;  
}  
}  
  
return 0;  
  
}
```

Java:

```
import java.util.*;  
  
import java.util.Arrays;  
  
class Main  
{  
  
    public static void display(int answer)
```



```

{
    System.out.println(answer);
}

//check char arrays

public static boolean CheckArrays(String original, String
duplicate)
{
    char a[]=original.toCharArray();
    char b[]=duplicate.toCharArray();
    int f=0;
    for(int i=0;i<original.length();i++)
    {
        if(a[i]==b[i])
            f+=1;
    }
    if(original.length()==f)
        return true;
    else
        return false;
}

//convert array to string

public static String ToString(char[] ch)

```

```
{  
  
    String ret="";  
    for(char c:ch)  
        ret+=c;  
  
    return ret;  
}  
  
public static void find(int numbers[],String original)  
{  
  
    String duplicate="";  
    char change[]=original.toCharArray();  
  
    int answer=1;  
    int loopings=0;  
    while(loopings==0)  
    {  
  
        char[] tc=new char[numbers.length];  
        for(int i=0;i<numbers.length;i++)  
        {  
  
            tc[numbers[i]]=change[i];  
  
        }  
  
        change=tc;  
  
        duplicate=ToString(change);  
    }  
}
```

```

        //System.out.println(original+"
"+duplicate);

        if(CheckArrays(original, duplicate)==true)

            loopings=1;

        else

            answer+=1;

    }

    display(answer);

}

public static void convert_to_char(int[] list)
{

    int length=list.length;

    char cha='a';

    char ch[]=new char[length];

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

        ch[i]=cha;

        cha++;

    }

    find(list,ToString(ch));

}

public static void main(String args[])

```

```
{  
  
    Scanner scan=new Scanner(System.in);  
  
    int t=scan.nextInt();  
  
    //int t=1;  
  
    if(t<1 || t>10)  
    {  
  
        System.exit(0);  
  
    }  
  
    for(int tc=0;tc<t;tc++)  
    {  
  
        int n=scan.nextInt();  
  
        //int n=6;  
  
        if(n<1 || n>10000)  
  
            System.exit(0);  
  
        int arr[]=new int[n];  
  
        for(int i=0;i<n;i++)  
        {  
  
            int x=scan.nextInt();  
  
            if(x<1 || x>n)  
  
                System.exit(0);  
  
            arr[i]=x-1;  
  
        }  
    }  
}
```

```

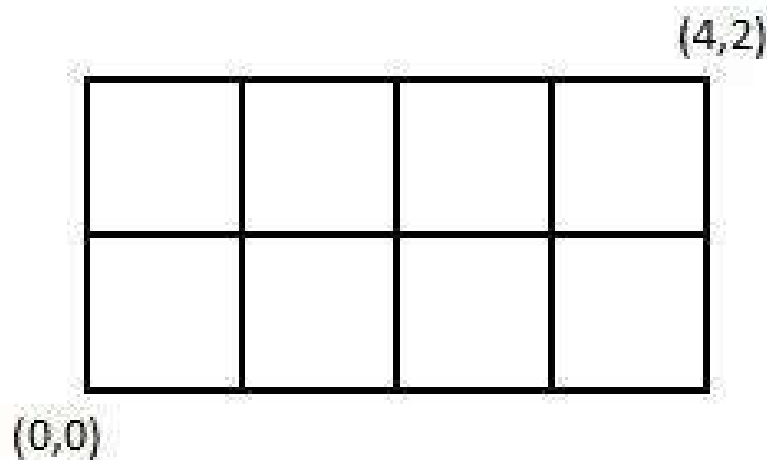
        convert_to_char(arr);
    }

}

}

```

4.



The city of Looneywalks has a rectangular grid of roads. One of the favorite pastimes of the folks at Looneywalks is to wander around the blocks. While they could take the shortest path, more often, they retrace their steps and engage themselves in finding the number of ways of reaching the destination with a given number of steps. For example, in the figure above a block is shown with 8 blocks and roads around them. Any walk is a series of symbols U, R, L, D where by U we mean moving one block towards north, R, moving one block East, L moving one block West and D moving one block South. One such path for the above reaching the upper right corner from lower left corner is RRRRUU.

This path has length 6 (the number of symbols in the path), which is the shortest path (that do not go outside the grid) between the two

points. If we enumerate all such shortest paths of length 6, there are 15 of them.

You are given the layout of the roads at Looneywalks. Given a starting point, S ending point E and the length k of the path, write a program to find the number of valid paths from S to E that have length precisely k. A valid path is one that does not go outside the grid, and two path is one

Every grid point is referred to by the X coordinate and a Y coordinate (the number of blocks east of the southwest corner, and the number of blocks north of the southwest corner). Thus the lower left corner in the above diagram is (0,0) and the top right corner is (4,2).

The size of the block is given by two numbers, M and N. M is the number of rows in the grid, and N is the number of columns (in the grid above, M is 2 and N is 4).

Constraints

$M, N < 10, k < 20$

Input

The first line of the input has three comma separated integers M, N and k. The Dimensions of the grid are M blocks in rows, and N blocks in columns. The path length is given by k.

The next line has four comma separated numbers giving the coordinates of the starting point and the ending point respectively.

Output

One integer representing the number of valid paths from the starting point to ending point that have length precisely k.

Explanation

Example 1

Input

2 4 6

0 0 4 2

Output

15

Explanation

M is 2, and N is 4. The value of k is 6, and we are counting paths of length 6. The starting point is (0,0) and the end point is (4,2).

The sketch of Loonewalks above is valid for this case. 6 is the length of the shortest path from the starting point to the ending point.

Hence, we need to find the number of shortest paths. Each such path has 4 moves to the right (R) and two moves up (U). All paths having 4 Rs and 2 Us are valid, as none of them will take the walker outside the grid. This is just the number of arrangements of 4 R and 2U in any order, which can be seen to be 15. Hence the result is 15.

Example 2

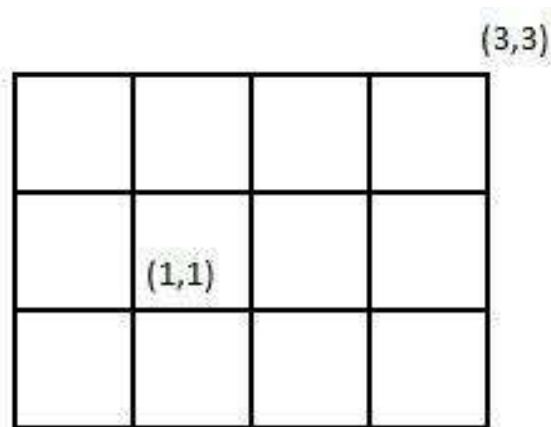
Input

3,3,6

1, 1, 3, 3

Output

90



Explanation

M=3, N=3,k=6. The starting and ending points are (1,1) and (3,3) respectively.

The shortest path is of length 4, and has two R and two U in the path. An example is RRUU. As the desired path length is 6, we need to add an additional R (and the corresponding L) to all the shortest paths, or add an additional U (and the corresponding D) to all the shortest paths.

Let us first consider adding one R and one L. The shortest path will have two R, and if we look at only the horizontal part of the path, the new sequence could be one of three, LRRR, RLRR or RRLR. Note that RRRL is not valid, as it will go outside the grid. We need to add two U to each of these. In say RLRR, there are 5 ways if the two Us are

together (UURLRR, RUULRR, and so on), and 10 ways if the Us are separate (URULRR, URLURR and so on), a total of 15 ways. Hence for all the three sequences, there are 45 paths if a RL pair is added. Similar considerations show that if we add a UD pair, there are another 45 paths. The total number of paths with length exactly 6 is therefore $45 + 45 = 90$, which is the result.

Solution:

C:

```
#include<stdio.h>
```

```
int path(int i,int j,int n,int m,int dn,int dm,int d){
    if(d<0) return 0;
    if(i==dn&& j==dm&& d==0) return 1;
    if(i<0 || j<0 || i>n || j>m) return 0; //base cases
    int sum=0;
    sum+=path(i+1,j,n,m,dn,dm,d-1); // going east
    sum+=path(i-1,j,n,m,dn,dm,d-1); // going west
    sum+=path(i,j+1,n,m,dn,dm,d-1); // going north
    sum+=path(i,j-1,n,m,dn,dm,d-1); // going south
    return sum;
}
```



```
signed main(){
    int n,m,k;
    scanf("%d %d %d",&n,&m,&k);
    int i,j,dn,dm;
    scanf("%d %d %d %d",&i,&j,&dn,&dm);
    printf("%d",path(i,j,n,m,dn,dm,k));
}
```

Java:

```
import java.util.*;

public class Main{

    public static int path(int i,int j,int n,int m,int dn,int dm,int d){
        if(d<0) return 0;
        if(i==dn&&j==dm&&d==0) return 1;
        if(i<0 || j<0 || i>n || j>m) return 0; //base cases
        int sum=0;
        sum+=path(i+1,j,n,m,dn,dm,d-1); // going east
        sum+=path(i-1,j,n,m,dn,dm,d-1); // going west
        sum+=path(i,j+1,n,m,dn,dm,d-1); // going north
        sum+=path(i,j-1,n,m,dn,dm,d-1); // going south
        return sum;
    }
}
```

```
}
```

```
public static void main(String[] args){
    int n,m,k;
    Scanner ip=new Scanner(System.in);
    n=ip.nextInt();
    m=ip.nextInt();
    k=ip.nextInt();
    int i,j,dn,dm;
    i=ip.nextInt();
    j=ip.nextInt();
    dn=ip.nextInt();
    dm=ip.nextInt();
    System.out.printf("%d",path(i,j,n,m,dn,dm,k));
}
}
```

Python:

```
def path(i,j,n,m,dn,dm,d):
    if d<0: return 0
    if(i==dn and j==dm and d==0): return 1;
    if(i<0 or j<0 or i>n or j>m): return 0;
    sum=0;
    sum+=path(i+1,j,n,m,dn,dm,d-1);
    sum+=path(i-1,j,n,m,dn,dm,d-1);
```

```

sum+=path(i,j+1,n,m,dn,dm,d-1);
sum+=path(i,j-1,n,m,dn,dm,d-1);
return sum;
n,m,k = list(map(int,(input().split(' '))))
i,j,dn,dm = list(map(int,(input().split(' '))))
print(path(i,j,n,m,dn,dm,k))

```

5. The task is to find the minimum sum of Products of two arrays of the same size, given that k modifications are allowed on the first array. In each modification, one array element of the first array can either be increased or decreased by 2.

Note- the product sum is Summation ($A[i]*B[i]$) for all i from 1 to n where n is the size of both arrays

Input Format:

First line of the input contains n and k delimited by whitespace

Second line contains the Array A (modifiable array) with its values delimited by spaces

Third line contains the Array B (non-modifiable array) with its values delimited by spaces

Output Format:

Output the minimum sum of products of the two arrays

Constraints:

$$1 \leq N \leq 10^5$$

$$0 \leq |A[i]|, |B[i]| \leq 10^5$$

$$0 \leq K \leq 10^9$$

Example

Input

3 5

1 2 -3

-2 3 -5

Output

-31

Explanation

Here total numbers are 3 and total modifications allowed are 5. So we modified A[2], which is -3 and increased it by 10 (as 5 modifications are allowed). Now final sum will be

$$(1 * -2) + (2 * 3) + (7 * -5)$$

$$-2 + 6 - 35$$

-31

-31 is our final answer.

Example**Input**

5 3

2 3 4 5 4

3 4 2 3 2

Output

25

Explanation

Here total numbers are 5 and total modifications allowed are 3. So we modified A[1], which is 3 and decreased it by 6 (as 3 modifications are allowed).

Now final sum will be

$$(2 * 3) + (-3 * 4) + (4 * 2) + (5 * 3) + (4 * 2)$$

$$6 - 12 + 8 + 15 + 8$$

25

25 is our final answer.

Solution:**C:**

```
#include<stdio.h>
```

```
#include<math.h>
```

```
#include<stdlib.h>
```

```
int minproduct(int a[], int b[], int n, int k)
{
    int diff = 0, res = 0;

    int temp;

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

        int pro = a[i] * b[i];

        res = res + pro;

        if (pro < 0 && b[i] < 0)
            temp = (a[i] + 2 * k) * b[i];
        else if (pro < 0 && a[i] < 0)
            temp = (a[i] - 2 * k) * b[i];
        else if (pro > 0 && a[i] < 0)
            temp = (a[i] + 2 * k) * b[i];
        else if (pro > 0 && a[i] > 0)
            temp = (a[i] - 2 * k) * b[i];

        int d = abs(pro - temp);

        if (d > diff)
            diff = d;
    }
}
```

```
        return res - diff;
    }

    int main()
    {
        int n,k,a[100],b[100];

        scanf("%d %d", &n, &k);

        for(int i = 0; i < n; i++)
            scanf("%d",&a[i]);

        for(int i = 0; i < n; i++)
            scanf("%d",&b[i]);

        /*    cout << minproduct(a, b, n, k)
               << endl; */

        printf("%d",minproduct(a, b, n, k));

        return 0;
    }
```

JAVA:

```
import java.util.Scanner;

import java.math.*;
```

```
class MinProduct {  
  
    static int minproduct(int a[], int b[], int n, int k)  
    {  
        int diff = 0, res = 0;  
  
        int temp = 0;  
  
        for (int i = 0; i < n; i++) {  
            int pro = a[i] * b[i];  
  
            res = res + pro;  
  
            if (pro < 0 && b[i] < 0)  
                temp = (a[i] + 2 * k) * b[i];  
            else if (pro < 0 && a[i] < 0)  
                temp = (a[i] - 2 * k) * b[i];  
            else if (pro > 0 && a[i] < 0)  
                temp = (a[i] + 2 * k) * b[i];  
            else if (pro > 0 && a[i] > 0)  
                temp = (a[i] - 2 * k) * b[i];  
  
            int d = Math.abs(pro - temp);  
  
            if (d > diff)
```

```
        diff = d;

    }

    return res - diff;

}

public static void main(String[] args)
{
    Scanner sc = new Scanner(System.in);

    int n = sc.nextInt();

    int k = sc.nextInt();

    int a[] = new int[n], b[] = new int[n];

    for(int i = 0; i < n ; i++)

        a[i] = sc.nextInt();

    for(int i = 0; i < n ; i++)

        b[i] = sc.nextInt();

    System.out.println(minproduct(a, b, n, k));

}

}
```

PYTHON:

```
def minproduct(a,b,n,k):
```



```

diff = 0

res = 0

for i in range(n):

    pro = a[i] * b[i]

    res = res + pro

    if (pro < 0 and b[i] < 0):

        temp = (a[i] + 2 * k) * b[i]

    elif (pro < 0 and a[i] < 0):

        temp = (a[i] - 2 * k) * b[i]

    elif (pro > 0 and a[i] < 0):

        temp = (a[i] + 2 * k) * b[i]

    elif (pro > 0 and a[i] > 0):

        temp = (a[i] - 2 * k) * b[i]

    d = abs(pro - temp)

    if (d > diff):

        diff = d

return res - diff

```

```
n,k = [int(x) for x in input().split()]
```

```
a = []
```

```
b = []
```

```
for i in range(n):  
    ele1 = int(input())  
    a.append(ele1)  
for i in range(n):  
    ele2 = int(input())  
    b.append(ele2)  
print(minproduct(a, b, n, k))
```

STRINGS BASED CODING PROBLEMS

1. In the Byteland country a string "S" is said to super ascii string if and only if count of each character in the string is equal to its ascii value.

In the Byteland country ascii code of 'a' is 1, 'b' is 2 ... 'z' is 26.

Your task is to find out whether the given string is a super ascii string or not.

Input Format:

First line contains number of test cases T, followed by T lines, each containing a string "S".

Output Format:

For each test case print "Yes" if the String "S" is super ascii, else print "No"

Constraints:

$1 \leq T \leq 100$

$1 \leq |S| \leq 400$, S will contains only lower case alphabets ('a'-'z').

Example 1

Input:

2

bba

scca

Output:

Yes

No

Explanation:

In case 1, viz. String "bba" -

The count of character 'b' is 2. Ascii value of 'b' is also 2.

The count of character 'a' is 1. Ascii value of 'a' is also 1.

Hence string "bba" is super ascii.

Solution:**C:**

```
#include<stdio.h>

#include<string.h>

int main()

{

int test, flag=0,length=0;

int j,count[26]={0,};

char arr[400];

scanf("%d",&test);

if(test>0 && test<=100)

{

while(test)

{

scanf("%s",arr);

length=strlen(arr);

if(length<=400)

{

for(j=0;arr[j]!='\0';j++)

{

if(arr[j]<97 || arr[j]>122)

{
```

```
    flag++;  
    break;  
}  
    count[(arr[j]-97)]++;  
}  
for(j=0;j<26;j++)  
{  
    if(count[j]!=0)  
    {  
        if(count[j]!=j+1)  
        {  
            flag++;  
            break;  
        }  
    }  
}  
if(flag)  
    printf("No");  
else  
    printf("Yes");  
}  
for(j=0;j<26;j++)
```

```
{
count[j]=0;
}
flag=0;
test--;
}
}
return 0;
}
```

Java:

```
import java.io.*;

class Main
{
    static int count[]=new int[27];

    public static void main(String args[])throws IOException
    {
        try
        {
            BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));

            System.out.println("Enter the No of Test case");

            int no=Integer.parseInt(br.readLine());
```

```
if(no>=1 && no<=100)
{
while(no!=0)
{
    System.out.println("Enter the String");
    String str=br.readLine();
    int len=str.length();
    if(strcheck(str,len))
    {
        for(int i=0;i<len;i++)
        {
            int value=((int)str.charAt(i))-96;
            count[value]+=1;
        }
        if(check())
        {
            System.out.println("YES");
        }
        else
        {
            System.out.println("NO");
        }
    }
}
```

```
        no=no-1;
    }
    else
    {
        System.out.println("Wrong String--
>1<=|S|<=400, S will contains only lower case alphabets ('a'-'z').\nre-
enter");
    }
}

else
{
    System.out.println("Test case Exceeded");
}
}
catch(Exception e)
{
}
}
```



```
public static boolean check()
{
    for(int i=1;i<=26;i++)
    {
        if(count[i]!=i && count[i]!=0)
        {
            return false;
        }
    }
    return true;
}

public static boolean strcheck(String str,int len)
{
    if(len<=1 || len >=400)
    {
        return false;
    }
    else
    {
        for(char c : str.toCharArray()) {
```

```

        if(Character.isLetter(c) &&
Character.isUpperCase(c)) {

                                return false;

        }

    }

}

return true;

}

}

```

Python:

```

n=int(input())

i=0

list=list()

while i<n:

    list.append(input())

    i=i+1

i=0

while i<n :

    s=list[i]

    j=0

    flag=1

```

```

while j<s.__len__() and flag==1:
    if s.count(s[j])!=(ord(s[j])-96):
        flag=0;
    j=j+1
if(flag==1):
    print("Yes")
else:
    print("No")
i=i+1

```

2. Rotate a given String in the specified direction by specified magnitude.

After each rotation make a note of the first character of the rotated String, After all rotation are performed the accumulated first character as noted previously will form another string, say

FIRSTCHARSTRING.

Check If **FIRSTCHARSTRING** is an Anagram of any substring of the Original string.

If yes print "YES" otherwise "NO". Input

The first line contains the original string s. The second line contains a single integer q. The ith of the next q lines contains character d[i] denoting direction and integer r[i] denoting the magnitude.

Constraints

1 <= Length of original string <= 30

1<= q <= 10

Output

YES or NO

Explanation

Example 1

Input

carrace

3

L 2

R 2

L 3

Output

NO

Explanation

After applying all the rotations the **FIRSTCHARSTRING** string will be "rcr" which is not anagram of any sub string of original string "carrace".

Solution:**C:**

```
#include<stdio.h>
```

```
int isSubstring(string s1, string s2)
```

```
{
```

```
    int M = strlen(s1);
```

```
    int N = strlen(s2);
```

```
    for (int i = 0; i <= N - M; i++) {
```

```
        int j;
```

```
        for (j = 0; j < M; j++)
```

```
            if (s2[i + j] != s1[j])
```

```
        break;

    if (j == M)
        return i;
}

return -1;
}

int main(){
    string s;
    scanf("%d",&s)
    int sz=s.size();
    string fc;
    int q;
    scanf("%d",&q);
    int i=0;
    while(q--){
        char c;
        int m;
        scanf("%c,%d",&c,&m);
        if(c=='L'){
            i=(i+m)%sz;
```

```

    }

    if(c=='R'){

        i=(i-m)%sz;

    }

    fc.push_back(s[i]);

}

bool f=false;

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

    for(int j=i;j<sz;j++){

        if(isSubstring(fc,s.substr(i,j-i+1))>=0) {

            f=true;

        }

    }

}

if(f) printf("yes");

else printf("NO");

}

```

Java:

```

import java.util.Arrays;

import java.util.Scanner;

public class Main {

    static String sample;

```

```
static String s1="";
```

```
public static String leftrotate(String str, int d)
```

```
{
```

```
sample = str.substring(d) + str.substring(0, d);
```

```
char fl = sample.charAt(0);
```

```
s1=s1+String.valueOf(fl);
```

```
return sample;
```

```
}
```

```
public static void rightrotate(String str, int d)
```

```
{
```

```
sample=leftrotate(str, str.length() - d);
```

```
}
```

```
public static boolean Anagram(char[] str1, char[] str2)
```

```
{
```

```
int len1 = str1.length;
```

```
int len2 = str2.length;
```

```
if (len1 != len2)
```

```
return false;
```

```
Arrays.sort(str1);
```

```

Arrays.sort(str2);

for (int i = 0; i < len1; i++)
    if (str1[i] != str2[i])
        return false;

return true;
}

public static void main(String args[])
{
    String[] dir;
    int[] mag;

    Scanner s = new Scanner(System.in);
    Scanner sp = new Scanner(System.in);

    sample = s.nextLine();

    int q=s.nextInt();

    dir = new String[q];
    mag = new int[q];

    for(int x=0;x<q;x++)
    {
        dir[x]= sp.nextLine();
        mag[x]=s.nextInt();
    }

    for(int i=0;i<q;i++)

```



```
{  
    if(dir[i].equals("L"))  
    {  
        int temp=mag[i];  
        leftrotate(sample, temp);  
    }  
    if(dir[i].equals("R"))  
    {  
        int temp=mag[i];  
        rightrotate(sample, temp);  
    }  
}  
  
char[] ans1 = sample.toCharArray();  
char[] s2 = s1.toCharArray();  
  
if (Anagram(ans1, s2))  
    System.out.println("YES");  
else  
    System.out.println("NO");  
}
```

```
}
```

Python:

```
def ana(a,b):
    if len(a)<len(b):
        return False
    cb=CharCounts(b)
    for i in range(0,len(a)-len(b)+1):
        window=a[i:i+len(b)]
        cw=CharCounts(window)
def CharCounts(s):
    counts=dict()
    for char in s:
        if char not in counts:
            counts[char]=1
        else:
            counts[char]+=1
    return counts
word=input().strip()
first=[]
i=0
q=int(input())
for _ in range(q):
```

```

d,s=input().strip().split()

s=int(s)

if d=="L":

    i=(i+s)%len(word)

else:

    i=(i-s)%len(word)

first.append(word[i])

if ana(word,first):

    print("yes")

else:

    print("no")

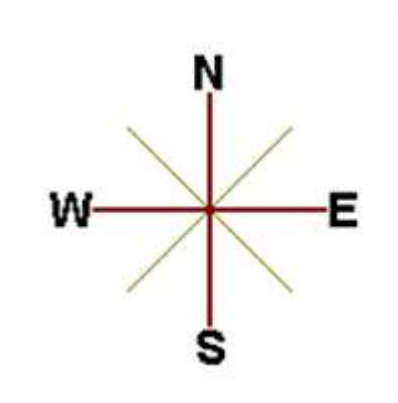
```

3. The amusement park at Patagonia has introduced a new skateboard competition. The skating surface is a grid of $N \times N$ squares. Most squares are so constructed with slopes that it is possible to direct the skateboard in any of up to three directions of the possible four (North ,East, South or West, represented by the letters N, E, S and W respectively). Some squares however have a deep drop from the adjacent square from which it is impossible to go to any adjacent square. These are represented by D (for Drop) in that square. The objective is to maneuver the skateboard to reach the South East corner of the grid, marked F.

Each contestant is given a map of the grid, which shows where the Drop squares are (marked D), where the Final destination is (marked F), and, for each other square, the directions it is possible to maneuver the skateboard in that square.

The contestant draws lots to determine which of the squares on the boundaries of the grid on the North or the West of the grid (the top or the left in the diagram) he or she should start in. Then, using a map of the grid, he or she needs to try to reach the South East corner destination by maneuvering the skateboard.

ES	ES	SE	ES	ES	S
SE	ES	SE	ES	ES	S
ES	ES	SE	ES	SE	S
ES	SE	ES	SE	E	D
SE	ES	D	WSE	NES	NS
E	E	NE	E	E	F



In some cases, it is impossible to reach the destination. For example, in the diagram above, if one starts at the North East corner (top right in the diagram), the only way is to go is South, until the Drop square is reached (three squares South), and the contestant is stuck there.

A contestant asks you to figure out the number of squares at the North or West boundary (top or left boundary in the map) from which it is feasible to reach the destination.

Constraints

$5 \leq N \leq 50$

Input Format

The first line of the input is a positive integer N , which is the number of squares in each side of the grid.

The next N lines have a N strings of characters representing the contents of the map for that corresponding row. Each string may be F , representing the Final destination, D , representing a drop square, or a set of up to three of the possible four directions (N, E, S, W) in some random order. These represent the directions in which the contestant can maneuver the skateboard when in that square.

Output

The output is one line with the number of North or West border squares from which there is a safe way to maneuver the skateboard to the final destination.

Explanation

Example 1

Input

```
6
ES,ES,SE,ES,ES,S
SE,ES,SE,ES,ES,S
ES,ES,SE,ES,SE,S
ES,SE,ES,SE,E,D
SE,ES,D,WSE,NES,NS
E,E,NE,E,E,F
```

Output

```
9
```

Explanation

$N = 6$, and the size of the grid is 6×6 . The map of the grid is as below.

ES	ES	SE	ES	ES	S
SE	ES	SE	ES	ES	S
ES	ES	SE	ES	SE	S
ES	SE	ES	SE	E	D
SE	ES	D	WSE	NES	NS
E	E	NE	E	E	F

There are many ways to maneuver the skateboard. For example, if the contestant starts from the North West corner (top left in the map) and goes East all the way until he reaches the top right corner in the map, and then goes South, he will reach a Drop square.

But if he goes South all the way from the same square until he reaches the bottom left square on the map, and keeps going East from there, the Final destination will be reached. Hence the top left square (1,1) is safe.

Similarly, from the square (1,5), all the paths lead to a drop square., The other 9 North and West border squares have ways skateboard to get to the final destination. The output is 9

Example 2

Input

5

ES,SE,ES,SE,S

S,EWS,SE,E,S

E,D,SEW,NSE,S

NE,N,SEW,D,W

EN,EN,E,EN,F

Output

4

Explanation

N=5, and the grid is 5 x 5 squares. The map of the grid looks like this.

ES	SE	ES	SE	S
S	EW	SE	E	S
E	D	SEW	NSE	S
NE	N	SEW	D	W
EN	EN	E	EN	F

It can be seen that from squares (1,4) and (1,5), there is no way to maneuver the skateboard to the Final destination, and hence are not safe starting points.. Similarly, squares (2,1),(3,1), and (4,1) are not safe starting points. The only safe starting points on the North and West borders are (1,1),(1,2),(1,3),(5,1). Hence the output is 4

ESESSEESS
 SEESSEESS
 ESESSEESS
 ESSEESSEED
 SEESDWSNESNS
 EENEEEF
 ESSEESSES
 SEWSSEES
 EDSEWNSES
 NENSEWDW
 ENENEENF

Solution:

C:

```

int ways(int A[][]) {

    if(A.size()==0) return 0;
  
```

```
int n=A.size();

int m=A[0].size();

vector<vector<int>> dp(n+1,vector<int>(m+1,1));

for(int i=0;i<n;i++) reverse(A[i].begin(),A[i].end());

reverse(A.begin(),A.end());

for(int i=1;i<=n;i++){

    for(int j=1;j<=m;j++){

        if(i==1)

            dp[i][j]=max(dp[i][j-1]-A[i-1][j-1],1);

        else if(j==1)

            dp[i][j]=max(dp[i-1][j]-A[i-1][j-1],1);

        else

            dp[i][j]=max(min(dp[i-1][j],dp[i][j-1])-A[i-1][j-1],1);

    }

}

return dp[n][m];

}

int main(){

int i,n,a[100];

scanf("%d",&n);

for(i=0;i<n;i++){
```



```
for(j=0;j<n;j++){  
    scanf("%d",&a[i][j]);  
}  
  
printf("%d",ways(a));
```

Java:

```
import java.util.*;  
  
private ArrayList<tangible.Pair<Integer,Integer>>[][] vc = new  
    ArrayList<tangible.Pair<Integer,Integer>>[10][10];  
  
private boolean[][] vis = new boolean[10][10];  
  
private int path(int i,int j,int n)  
{  
    if (i < 0 || j < 0 || i >= n || j >= n)  
    {  
        return 0;  
    }  
  
    if (i == n - 1 && j == n - 1)  
    {  
        return 1;  
    }  
}
```

```

vis[i][j] = true;

int mink = 0;

for (int ii = 0; ii < vc[i][j].size(); ii++)
{
    if (!vis[i + vc[i][j].get(ii).first][j + vc[i][j].get(ii).second])
    {
        mink = (mink | path(i + vc[i][j].get(ii).first, j +
vc[i][j].get(ii).second, n));
    }
}

vis[i][j] = false;

return mink;
}

private ArrayList<String> split(String str, char delimiter)
{
    ArrayList<String> internal = new ArrayList<String>();

    stringstream ss = new stringstream(str); // Turn the string into a
stream.

    String tok;

    while (getline(ss, tok, delimiter))
    {

        internal.add(tok);
    }
}

```

```

    }

    return new ArrayList<String>(internal);
}

private static int main()
{
    int n;

    n = Integer.parseInt(ConsoleInput.readToWhiteSpace(true));

to 'sizeof':

    memset(vis,false,sizeof(vis));

    ArrayList<ArrayList<String>> v = new
ArrayList<ArrayList<String>>();

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

        String s;

        s = ConsoleInput.readToWhiteSpace(true);

        ArrayList<String> sep = split(s, ',');

        v.add(sep);

    }

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

        for (int j = 0;j < n;j++)

            {

```

```
for (int k = 0;k < v.get(i).get(j).length();k++)
{
    if (v.get(i).get(j).charAt(k) == 'E')
    {
        vc[i][j].add({0,1});
    }
    if (v.get(i).get(j).charAt(k) == 'W')
    {
        vc[i][j].add({0,-1});
    }
    if (v.get(i).get(j).charAt(k) == 'N')
    {
        vc[i][j].add({-1,0});
    }
    if (v.get(i).get(j).charAt(k) == 'S')
    {
        vc[i][j].add({1,0});
    }
}
}
```

```
int ans = 0;

for (int i = 0; i < n; i++)
{
    ans += path(0, i, n);
}

for (int i = 1; i < n; i++)
{
    ans += path(i, 0, n);
}
```

```
System.out.print(ans);
```

```
public final class ConsoleInput
{
    private static boolean goodLastRead = false;

    public static boolean lastReadWasGood()
    {
        return goodLastRead;
    }

    public static String readToWhiteSpace(boolean
skipLeadingWhiteSpace)
```

```
{  
    String input = "";  
    char nextChar;  
    while (Character.isWhitespace(nextChar =  
(char)System.in.read()))  
    {  
        //accumulate leading white space if  
skipLeadingWhiteSpace is false:  
        if (!skipLeadingWhiteSpace)  
        {  
            input += nextChar;  
        }  
    }  
    input += nextChar;  
  
    while (!Character.isWhitespace(nextChar =  
(char)System.in.read()))  
    {  
        input += nextChar;  
    }  
  
    goodLastRead = input.length() > 0;  
    return input;
```

```
}
```

```
public static String scanfRead()
```

```
{
```

```
    return scanfRead(null, -1);
```

```
}
```

```
public static String scanfRead(String unwantedSequence)
```

```
{
```

```
    return scanfRead(unwantedSequence, -1);
```

```
}
```

```
public static String scanfRead(String unwantedSequence, int
maxFieldLength)
```

```
{
```

```
    String input = "";
```

```
    char nextChar;
```

```
    if (unwantedSequence != null)
```

```
    {
```

```
        nextChar = '\0';
```

```

        for (int charIndex = 0; charIndex <
unwantedSequence.length(); charIndex++)
        {
            if
(Character.isWhitespace(unwantedSequence.charAt(charIndex)))
            {

                while (Character.isWhitespace(nextChar
= (char)System.in.read()))
                {
                }

            }
            else
            {

                //ensure each character matches the
expected character in the sequence:

                nextChar = (char)System.in.read();

                if (nextChar !=
unwantedSequence.charAt(charIndex))

                    return null;

            }
        }
    }

```



```

        input = (new Character(nextChar)).toString();
        if (maxLength == 1)
            return input;
    }

    while (!Character.isWhitespace(nextChar =
(char)System.in.read()))
    {
        input += nextChar;
        if (maxLength == input.length())
            return input;
    }

    return input;
}
}

```

```

public final class Pair<T1, T2>
{
    public T1 first;
    public T2 second;
}

```

```
public Pair()
```

```
{  
    first = null;  
    second = null;  
}
```

```
public Pair(T1 firstValue, T2 secondValue)
```

```
{  
    first = firstValue;  
    second = secondValue;  
}
```

```
public Pair(Pair<T1, T2> pairToCopy)
```

```
{  
    first = pairToCopy.first;  
    second = pairToCopy.second;  
}  
}
```

4. Romeo and Juliet stay in Rectanglevilla. As the name suggests, Rectanglevilla is indeed a rectangle in shape. Romeo's current location is marked by 's'. Juliet's current location is marked by 'd'. In a

difficult hour, Romeo has to ensure that he reaches Juliet via the shortest path and hence in shortest time.

There are many in Rectangleville who are enemies of Romeo and hence will not allow Romeo to pass through their areas. Romeo must avoid these areas while reaching out for Juliet. These areas are marked by 'w' character. Areas that Romeo can freely access is marked by '-' character.

Today Romeo needs your help to reach Juliet via shortest path. Help him.

The input and output specification sections describe how inputs will be provided on console and how output is expected back on console.

Input Format:

1. First line of input contains grid dimensions delimited by space character
2. Next *rows* number of lines contain *column* number of characters
3. Valid characters are {s, d, w, -}, where
 - a.s represents the location of the Romeo
 - b.d represents the location of the Juliet
 - c.w represents inaccessible region
 - d.- represents accessible region
4. End of input is marked by -1 character

Output Format:

1. Output grid should be of the same dimension as the input grid

2. Output grid should contain path from Romeo's location s to Juliet's location d
3. The 's' character in the grid must be replaced by character 'a' to denote that Romeo is actively heading towards Juliet.

Constraints:

1. It is guaranteed that there will always be exactly one shortest path for Romeo to reach Juliet

Test Cases

Input

```
4 4
w - d -
w - w -
w - w w
s - - -
-1
```

Output

```
w - d -
w a w -
w a w w
a - - -
```

Test Cases

Input

```
6 6
s - - - - w
- - - - w -
- - - w - -
- - w - - -
- w - - - -
w - - - - d
-1
```

Output

```
a - - - - w
```

```
- a - - w -
- - a w - -
- - w a - -
- w - - a -
w - - - - d
```

Solution:

C:

```
#include<stdio.h>

signed main(){

    int n,m,k;

    scanf("%d %d",&n,&m);

    char ar[n][m];

    for(int i=0;i<n;i++)

        for(int j=0;j<m;j++)

            scanf("%c",&ar[i][j]);

    scanf("%d",&k);

    int arr[n][m];

    int inx,iny,fnx,fny,dir;

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

        for(int j=0;j<m;j++) {

            if(ar[i][j]=='w') arr[i][j]=0;

            else if(ar[i][j]=='-') arr[i][j]=1;

            else if(ar[i][j]=='s') { inx=i;iny=j;arr[i][j]=1;}
```

```

else if (ar[i][j]=='d') fnx=i,fny=j,arr[i][j]=1;

}

}

if(fnx>inx) dir=1;

else dir=-1;

int i=inx,j=iny;

arr[i][j]=2;

while(i!=fnx || j!=fny){

    if(i==fnx) arr[i][j+1]=2,j=j+1;

    else if(j==fny) arr[i+dir][j]=2,i=i+dir;

    else if(arr[i+dir][j+1]==1)
arr[i+dir][j+1]=2,i=i+dir,j=j+1;

    else if(arr[i+dir][j]==1) arr[i+dir][j]=2,i=i+dir;

    else if(arr[i][j+1]==1) arr[i][j+1]=2,j=j+1;

}

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

for(int j=0;j<m;j++){

    if(arr[i][j]==2) ar[i][j]='a';

    ar[fnx][fny]='d';

    printf("%c",ar[i][j]);

```

```
    }  
}
```

```
}
```

Java:

```
import java.io.BufferedReader;
```

```
import java.io.IOException;
```

```
import java.io.InputStreamReader;
```

```
import java.util.StringTokenizer;
```

```
public class Main {
```

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

```
        BufferedReader br = new BufferedReader(new  
        InputStreamReader(System.in));
```

```
        try {
```

```
            int numCases = Integer.parseInt(br.readLine());
```

```

while(numCases>0){

    StringTokenizer      st      =      new
StringTokenizer(br.readLine());

    int N = Integer.parseInt(st.nextToken());

    int K = Integer.parseInt(st.nextToken());

    st = new StringTokenizer(br.readLine());

    int      numRomeo      =
Integer.parseInt(st.nextToken());

    int[] workRomeo = new int[N+1];

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

        int      workingDay      =
Integer.parseInt(st.nextToken());

        workRomeo[workingDay] = 1;

    }

    st = new StringTokenizer(br.readLine());

```



```

        int numJuliet = Integer.parseInt(st.nextToken());

        int[] workJuliet = new int[N+1];

        for(int i=0; i<numJuliet; i++)
        {
            int workingDay =
Integer.parseInt(st.nextToken());

            workJuliet[workingDay] = 1;
        }

        int startDay = findStartingDay(N, K,
workRomeo, workJuliet);

        System.out.println(startDay);

        numCases--;
    }

} catch (NumberFormatException e) {

    e.printStackTrace();

} catch (IOException e) {

    e.printStackTrace();

}

```

```
}
```

```
private static int findStartingDay(int N, int K, int[] workRomeo,  
    int[] workJuliet) {
```

```
    int missRomeo[] = new int[N+1];
```

```
    int leavesJuliet[] = new int[N+1];
```

```
    for(int i=1;i<=N;i++){
```

```
        if(workRomeo[i] == 0)
```

```
            missRomeo[i] = missRomeo[i-1] + 1;
```

```
        else
```

```
            missRomeo[i] = missRomeo[i-1];
```

```
        leavesJuliet[i] = leavesJuliet[i-1] + workJuliet[i];
```

```
    }
```

```
    //System.out.println(Arrays.toString(missRomeo));
```

```
    //System.out.println(Arrays.toString(leavesJuliet));
```

```
    int start = 0, best=-1, leaves=Integer.MAX_VALUE;
```

```
int lim = N-K+1;
```

```
for(int i=1;i<=lim;i++) {
```

```
    int days=missRomeo[i+K-1]-missRomeo[i-1]; // no of
    days they can meet
```

```
    if(days>best)
```

```
    {
```

```
        best=days;
```

```
        start=i;
```

```
        leaves=leavesJuliet[i+K-1]-leavesJuliet[i-1]; //
    no of leaves juliet has to take
```

```
    }
```

```
    else if(days==best)
```

```
    {
```

```
        int tmp=leavesJuliet[i+K-1]-leavesJuliet[i-1];
```

```
        if(tmp<leaves)
```

```
        {
```

```
            leaves=tmp;
```

```
            start=i;
```

```
        }
```

```

    }

    }

    return start;

}

}

```

5. Smilodon is a ferocious animal which used to live during the Pleistocene epoch (2.5 mya–10,000 years ago). Scientists successfully created few smilodons in an experimental DNA research. A park is established and those smilodons are kept in a cage for visitors. This park consists of Grasslands(G), Mountains(M) and Waterbodies(W) and it has three gates (situated in grasslands only). Below is a sample layout.

W	M	G	G	G	G
M	G	W	G	M	M
G	G	G	G	G	G
W	G	G	M	W	G


Before opening the park, club authority decides to calculate Safety index of the park. The procedure of the calculation is described below. Please help them to calculate.

Safety Index calculation

Assume a person stands on grassland(x) and a Smilodon escapes from the cage situated on grassland(y). If the person can escape from any of those three gates before the Smilodon able to catch him, then the grassland(x) is called safe else it is unsafe. A person and a Smilodon both take 1 second to move from one area to another adjacent area(top, bottom, left or right) but a person can move only over grasslands though Smilodon can move over grasslands and mountains.

If any grassland is unreachable for Smilodon(maybe it is unreachable for any person also), to increase safe index value Club Authority use to mark those grasslands as safe land. Explained below

W	M	G	G	G	G
M	G	W	G(x)	M	M
G	W	G	G(y)	G	G
W	G(z)	W	M	W	G



For the above layout, there is only one gate at (4,6)

Y is the position of Smilodon's cage

X is not safe area

Z is a safe area as is it not possible for smilodon to reach z

Safety index=(total grassland areas which are safe*100)/total grassland area

Constraints

$3 \leq R, C \leq 10^3$

Gates are situated on grasslands only and at the edge of the park

The cage is also situated in grassland only

The position of the cage and the position of three gates are different

Input Format

The first line of the input contains two space-separated integers R and C, denoting the size of the park ($R \times C$)

The second line contains eight space-separated integers where

First two integers represent the position of the first gate

3rd and 4th integers represent the position of second gate

5th and 6th integers represent the position of third gate respectively

The last two integers represent the position of the cage

Next R lines, each contains space separated C number of characters.

These R lines represent the park layout.

Output

Safety Index accurate up to two decimal places using Half-up

Rounding method

Explanation

Example 1

Input

4 4

1 1 2 1 3 1 1 3

G G G G

G W W M

G G W W

M G M M

Output

75.00

Explanation

G	G	G	G
G	W	W	M
G	G	W	W
M	G	M	M
Mountains		4	
Gates- Safe Areas		3	
Other Safe Areas		3	
Waters		4	
Cage Pos.-unsafe		1	
Other unsafe areas		1	

Safety Index= (6*100)/8

Example 2

Input

4 6

1 6 3 6 4 6 3 4

W M G G G G

M G W G M M

G W G G G G

W G W M W G

Output

69.23

Solution:

C:

```
#include<stdio.h>

int A[50][50]={0};

int vis[50][50];

int path(int i,int j,int n,int dn,int dm){

    vis[i][j]=1;

    if(i<0 || j<0 || i>=n || j>=n) return 0;

    if(i==dn&& j==dm) return 1;

    int sum=0;

    for(int ii=0;ii<5;ii++){

        if(!vis[i][j-1]&&A[i][j]) sum+=path(i,j-1,n,dn,dm);

        if(!vis[i+1][j]&&A[i][j]) sum+=path(i+1,j,n,dn,dm);

        if(!vis[i-1][j]&&A[i][j]) sum+=path(i-1,j,n,dn,dm);

        if(!vis[i][j+1]&&A[i][j]) sum+=path(i,j+1,n,dn,dm);

    }

    vis[i][j]=0;

    return sum;

}

signed main(){

    int n,m;

    scanf("%d %d",&n,&m);
```



```
int x1,y1,x2,y2,x3,y3,x4,y4;

scanf("%d%d%d%d%d%d%d%d",&x1,&y1,&x2,&y2,&x3,&y3,&x
4,&y4);

int count=0;

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

    for(int j=0;j<m;j++){

        char c;

        scanf("%c",&c);

        if(c=='G') A[i][j]=1,count++;

    }

}

x4--;y4--;

A[x4][y4]=0;


int g=0;

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

    for(int j=0;j<m;j++){

        if(A[i][j]){

            if(path(i,j,n,x4,y4)) A[i][j]=0;

            if(A[i][j]) g++;

        }

    }

}
```

```
    }  
}  
  
printf("%f", (float)(100*(g/10?g-2:g))/(float)count);  
  
}
```

Java:

```
package demo;
```

```
import java.util.Scanner;
```

```
public class DemoTranslation {
```

```
    public static int[][] a = new int[50][50];
```

```
    public static int[][] vis = new int[50][50];
```

```
    public static int path(int i, int j, int n, int dn, int dm) {
```

```
        vis[i][j] = 1;
```

```
        if(i < 0 || j < 0 || i >= n || j >= n) {
```

```
            return 0;
```

```
        }
```

```
        if(i == dn && j == dm) {
```

```
        return 1;
    }

    int sum = 0;

    for(int ii = 0; ii < 5; ii++) {

        if(vis[i][j - 1] == 0 && a[i][j] != 0) {

            sum |= path(i, j - 1, n, dn, dm);

        }

        if(vis[i + 1][j] == 0 && a[i][j] != 0) {

            sum |= path(i + 1, j, n, dn, dm);

        }

        if(vis[i - 1][j] == 0 && a[i][j] != 0) {

            sum |= path(i - 1, j, n, dn, dm);

        }

        if(vis[i][j + 1] == 0 && a[i][j] != 0) {

            sum |= path(i, j + 1, n, dn, dm);

        }

    }

    vis[i][j] = 0;

    return sum;

}
```

```
public static void main(String[] args) {  
    int n, m;  
  
    n = STDIN_SCANNER.nextInt();  
    m = STDIN_SCANNER.nextInt();  
  
    int x1, y1, x2, y2, x3, y3, x4, y4;  
  
    x1 = STDIN_SCANNER.nextInt();  
    y1 = STDIN_SCANNER.nextInt();  
    x2 = STDIN_SCANNER.nextInt();  
    y2 = STDIN_SCANNER.nextInt();  
    x3 = STDIN_SCANNER.nextInt();  
    y3 = STDIN_SCANNER.nextInt();  
    x4 = STDIN_SCANNER.nextInt();  
    y4 = STDIN_SCANNER.nextInt();  
  
    int count = 0;  
  
    for(int i = 0; i < n; i++) {  
        for(int j = 0; j < m; j++) {  
            byte c;  
  
            c = (byte)nextChar(STDIN_SCANNER);  
  
            if(c == 'G') {  
                a[i][j] = 1;  
            }  
        }  
    }  
}
```

```

        count++;
    }
}

}

x4--;

y4--;

a[x4][y4] = 0;


int g = 0;

for(int i = 0; i < n; i++) {
    for(int j = 0; j < m; j++) {
        if(a[i][j] != 0) {
            if(path(i, j, n, x4, y4) != 0) {
                a[i][j] = 0;
            }
            if(a[i][j] != 0) {
                g++;
            }
        }
    }
}
}

```

```

        System.out.printf("%f", (float)(100 * (g / 10 != 0 ? g - 2 : g))
/ (float)count);

    }

```

```

    public final static Scanner STDIN_SCANNER = new
Scanner(System.in);

```

```

/**
 * This method is missing from the Scanner interface.
 */

public final static int nextChar(Scanner scanner) {

    scanner.useDelimiter("");

    int ret = scanner.next().charAt(0);

    scanner.reset();

    return ret;

}

```

```

}

```

Python:

```

for i in range(int(input())):

    n,m,q,p,r,k = map(int,input().split())

    done = 0

    rides = []

```

```
for clock in range(k+1):  
    while(len(rides)):  
        if clock-rides[0]==p:  
            rides.pop(0)  
            done+=1  
            n+=1  
        else:  
            break  
    if m and clock%r == 0 and clock:  
        m-=1  
        q+=1  
    while n and q:  
        n-=1  
        q-=1  
        rides.append(clock)  
print(n,done,m,q)
```

MATRIX BASED QUESTIONS

1. Sam is an eligible bachelor. He decides to settle down in life and start a family. He goes bride hunting.

He wants to marry a girl who has at least one of the 8 qualities mentioned below:-

- 1) The girl should be rich.
- 2) The girl should be an Engineer/Doctor.
- 3) The girl should be beautiful.
- 4) The girl should be of height 5.3".
- 5) The girl should be working in an MNC.
- 6) The girl should be an extrovert.
- 7) The girl should not have spectacles.
- 8) The girl should be kind and honest.

He is in search of a bride who has some or all of the 8 qualities mentioned above. On bride hunting, he may find more than one contenders to be his wife.

In that case, he wants to choose a girl whose house is closest to his house. Find a bride for

Sam who has maximum qualities. If in case, there are more than one contenders who are at equal distance from Sam's house; then print ""Polygamy not allowed"".

In case there is no suitable girl who fits the criteria then print ""**No suitable girl found**""

Given a Matrix $N \times M$, Sam's house is at (1, 1). It is denoted by 1. In the same matrix, the location of a marriageable Girl is also denoted by 1. Hence 1 at location (1, 1) should not be considered as the location of a marriageable Girl's location.

The qualities of that girl, as per Sam's criteria, have to be decoded from the number of non-zero neighbors (max 8-way) she has. Similar to the condition above, 1 at location (1, 1) should not be considered as the quality of a Girl. See Example section to get a better understanding.

Find Sam, a suitable Bride and print the row and column of the bride, and find out the number of qualities that the Bride possesses.

NOTE: - Distance is calculated in number of hops in any direction i.e. (Left, Right, Up, Down and Diagonal)

Constraints

$2 \leq N, M \leq 10^2$

Input Format

First Line contains the row (N) and column (M) of the houses.

Next N lines contain the data about girls and their qualities.

Output

It will contain the row and column of the bride, and the number of qualities that Bride possess separated by a colon (i.e. :).

Explanation

Example 1

Input:

```
2 9
1 0 1 1 0 1 1 1 1
0 0 0 1 0 1 0 0 1
```

Output:

```
1:7:3
```

Explanation:

The girl and qualities are present at

(1,3),(1,4),(1,6),(1,7),(1,8),(1,9),(2,4),(2,6),(2,9).

The girl present at (1,3) has 2 qualities (i.e. (1,4) and (2,4)).

The girl present at (1,4) has 2 qualities.

The Bride present at (1,6) has 2 qualities.

The Bride present at (1,7) has 3 qualities.

The Bride present at (1,8) has 3 qualities.

The Bride present at (1,9) has 2 qualities.

The Bride present at (2,4) has 2 qualities.

The Bride present at (2,6) has 2 qualities.

The Bride present at (2,9) has 2 qualities.

As we see, there are two contenders who have maximum qualities, one is at (1,7) and another at (1,8).

The girl who is closest to Sam's house is at (1,7). Hence, she is the bride.

Hence, the output will be 1:7:3.

Example 2

Input:

```
6 6
1 0 0 0 0 0
0 0 0 0 0 0
0 0 1 1 1 0
0 0 1 1 1 0
0 0 1 1 1 0
0 0 0 0 0 0
```

Output:

```
4:4:8
```

Explanation:

The bride and qualities are present at

(3,3),(3,4),(3,5),(4,3),(4,4),(4,5),(5,3),(5,4),(5,5)

The Bride present at (3,3) has 3 qualities (i.e. (3,4),(4,3) and (4,4)).

The Bride present at (3,4) has 5 qualities.

The Bride present at (3,5) has 3 qualities.

The Bride present at (4,3) has 5 qualities.

The Bride present at (4,4) has 8 qualities.

The Bride present at (4,5) has 5 qualities.

The Bride present at (5,3) has 3 qualities.

The Bride present at (5,4) has 5 qualities.

The Bride present at (5,5) has 3 qualities.

As we see, the girl present in (4,4) has maximum number of Qualities. Hence, she is the bride.

Hence, the output will be 4:4:8.

Solution:

C:

```
#include<stdio.h>
```

```
int main()
```

```
{
```

```
int
n,m,i,g[50][50],j,p,q,max=0,cnt=0,k=1,c=0,u=1,x[30],y[30],t1,min=0,
sc[50],e,f,ct=0,a[50],count=0,t2=0,t=0;

scanf("%d %d",&n,&m);

for(i=1;i<=n;i++)
{
for(j=1;j<=m;j++)
{
scanf("%d",&g[i][j]);
}
}

g[1][1]=0;

for(i=1;i<=n;i++)
{
for(j=1;j<=m;j++)
{
cnt=0;

if(g[i][j]==1)
{
t++;

for(p=i-1;p<=i+1;p++)
{
```

```
for(q=j-1;q<=j+1;q++)
{

    if(g[p][q]==1)
    {
        cnt++;
    }
}

}cnt=cnt-1;

a[k]=cnt;

k++;

}

}

for(k=1;k<=t;k++)
{

    if(a[k]>max)

        max=a[k];

}

if(max==0)

{

    printf("No suitable girl found");
```

```
goto x;

}

for(k=1;k<=t;k++)

{

    if(a[k]==max)

        c++;

}

for(k=1;k<=t;k++)

{

    t2=0;

    if(a[k]==max)

    {

        for(i=1;i<=n;i++)

        {

            for(j=1;j<=m;j++)

            {

                if(g[i][j]==1)

                    t2++;

                if(t2==k)

                {

                    x[u]=i;

                    y[u]=j;
```

```
        u++;
    }
}
}
}
}
}
t1=u-1;
if(c==1)
printf("%d:%d:%d",x[1],y[1],max);
else
{
    for(u=1;u<=t1;u++)
    {
        e=x[u]-1;
        f=y[u]-1;
        if(e>=f)
        {
            sc[u]=e;
        }
        else
            sc[u]=f;
    }
}
```

```
min=sc[1];
for(u=1;u<=t1;u++)
{
    if(sc[u]<min)
        min=sc[u];
}
for(u=1;u<=t1;u++)
{
    if(sc[u]==min)
        count++;
}
if(count>1)
    printf("Polygamy not allowed");
if(count==1)
{
    for(u=1;u<=t1;u++)
    {
        if(sc[u]==min)
            printf("%d:%d:%d",x[u],y[u],max);
    }
}
```

```
    }  
    x: return 0;  
}
```

Java:

```
import java.util.Scanner;  
import java.util.ArrayList;  
import java.util.Collections;  
  
class Main{  
  
    public int[][] arr;  
  
    class Data{  
  
        int quality;  
  
        int rowno;  
  
        int columnno;  
  
        Data(int q,int i,int j){  
  
            this.quality = q+1;  
  
            this.rowno = i+1;  
  
            this.columnno = j+1;  
  
        }  
  
    }  
  
    public static void main(String args[]){  
  
        Scanner sc = new Scanner(System.in);  
  
        Main b = new Main();
```



```

ArrayList<Data> list = new ArrayList<>();

ArrayList<Integer> max = new ArrayList<>();

int rows = sc.nextInt();

int column = sc.nextInt();

b.arr = new int[rows][column];

for(int i=0;i<rows;i++){
    for(int j=0;j<column;j++){
        b.arr[i][j] = sc.nextInt();
    }
}

for(int i=0;i<rows;i++){
    for(int j=0;j<column;j++){
        if(1==b.arr[i][j]){

            int qualities = b.qualities(i,j,b.arr);

            list.add(b.new Data(qualities,i,j));

            int x=i;

            int y=j;

            max.add(qualities);

            x++;

            y++;

            System.out.println("Qualitites at the position "+"("+x+", "+y+"
            "+"is"+ qualities);

```

```

    }

    }

    }

    int maximum = Collections.max(max);

    Data d = list.get(max.indexOf(maximum));

    System.out.println(d.rowno+":"+d.columnno":"+maximum);

    }

    int qualities(int i,int j,int[][] arr){

    int total=0;

    try{

    if(arr[i][j+1]==1){

    total++;

    }

    }catch(Exception e){}

    try{

    if(arr[i][j-1]==1){

    total++;

    }

    }catch(Exception e){

    }

    try{

```

```
if(arr[i+1][j+1]==1){  
    total++;  
}  
}catch(Exception e){  
}  
try{  
    if(arr[i-1][j+1]==1){  
        total++;  
    }  
}catch(Exception e){  
}  
try{  
    if(arr[i-1][j-1]==1){  
        total++;  
    }  
}catch(Exception e){  
}  
try{  
    if(arr[i+1][j-1]==1){  
        total++;  
    }  
}catch(Exception e){
```

```

    }

    try{

        if(arr[i+1][j]==1){
            total++;
        }

    }catch(Exception e){

    }

    try{

        if(arr[i-1][j]==1){
            total++;
        }

    }catch(Exception e){

    }

    return total;

}

}

```

Python:

```

def operation(n):

    if n == 0:

        return False

```

```
else :  
    return True  
  
def findqualities(mat, p1, p2, row, col):  
    count = 0  
    try:  
        if p2+1 <=col and mat[p1][p2+1] == 1:  
            count += 1  
    except:  
        pass  
    try:  
        if p2-1 >=0 and mat[p1][p2-1] == 1:  
            count += 1  
    except :  
        pass  
    try:  
        if p1-1>=0 and mat[p1-1][p2] == 1:  
            count += 1  
    except :  
        pass  
    try:  
        if p1+1 <=row and mat[p1+1][p2] == 1:  
            count += 1
```

```

except:
    pass

try:
    if mat[p1 - 1][p2 + 1] == 1:
        count += 1

except : pass

try:
    if mat[p1 + 1][p2 - 1] == 1:
        count += 1

except :
    pass

try:
    if mat[p1 + 1][p2 + 1] == 1:
        count += 1

except:pass

try:
    if mat[p1 - 1][p2 - 1] == 1:
        count += 1

except:
    pass

return count

row = int(input())

```

```
col = int(input())

input_mat = [ list(int(a) for a in input().split()) for _ in range(row)]

a = []

input_mat[0][0]=0

for pos1, lis in enumerate(input_mat,0):

    for pos2, ele in enumerate(lis,0):

        if operation(ele):

            y = findqualities(input_mat, pos1, pos2, row, col)

            a.append([pos1,pos2,y])

max = a[0][2]

max_pos = a[0][1], a[0][2]

for pos1, ele in enumerate(a):

    if ele[2] > max:

        max = ele[2]

        max_pos = [[ele[0], ele[1]], ]

for pos1, ele in enumerate(a):

    if ele[2] == max:

        max_pos.append([ele[0], ele[1]])

del(max_pos[0])

print(str(max_pos[0][0]+1) + ":" + str(max_pos[0][1]+1) + ":" +
str(max-1))
```

2. During the battle of Mahabharat, when Arjuna was far away in the battlefield, Guru Drona made a Chakravyuha formation of the Kaurava army to capture Yudhisthir Maharaj. Abhimanyu, young son of Arjuna was the only one amongst the remaining Pandava army who knew how to crack the Chakravyuha. He took it upon himself to take the battle to the enemies.

Abhimanyu knew how to get power points when cracking the Chakravyuha. So great was his prowess that rest of the Pandava army could not keep pace with his advances. Worried at the rest of the army falling behind, Yudhisthir Maharaj needs your help to track of Abhimanyu's advances. Write a program that tracks how many power points Abhimanyu has collected and also uncover his trail

A Chakravyuha is a wheel-like formation. Pictorially it is depicted as below

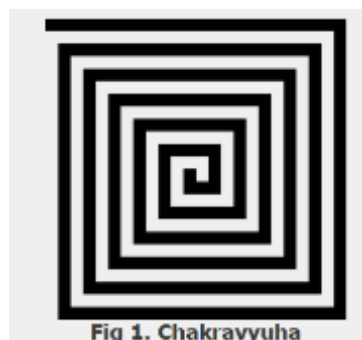


Fig 1. Chakravyuha

A Chakravyuha has a very well-defined co-ordinate system. Each point on the co-ordinate system is manned by a certain unit of the army. The Commander-In-Chief is always located at the center of the army to better co-ordinate his forces. The only way to crack the Chakravyuha is to defeat the units in sequential order.

A Sequential order of units differs structurally based on the radius of the Chakra. The radius can be thought of as length or breadth of the matrix depicted above. The structure i.e. placement of units in *sequential order* is as shown below

1	2	3	4	5
16	17	18	19	6
15	24	25	20	7
14	23	22	21	8
13	12	11	10	9

Fig 2. Army unit placements in Chakravyuha of size 5

The entry point of the Chakravyuha is always at the (0,0) co-ordinate of the matrix above. This is where the 1st army unit guards. From (0,0) i.e. 1st unit Abhimanyu has to march towards the center at (2,2) where the 25th i.e. the last of the enemy army unit guards.

Remember that he has to proceed by destroying the units in sequential fashion. After destroying the first unit, Abhimanyu gets a power point. Thereafter, he gets one after destroying army units which are multiples of 11. You should also be in a position to tell Yudhisthir Maharaj the location at which Abhimanyu collected his power points.

Input Format: First line of input will be length as well as breadth of the army units, say N

Output Format:

Print NxN matrix depicting the placement of army units, with unit numbers delimited by (\t) Tab character

Print Total power points collected

Print coordinates of power points collected in sequential fashion (one per line)

Constraints: $0 < N \leq 100$

Test Cases

Input

2

Output

1 2

3 4

Total power point:1

(0,0)

Input

5

Output

1 2 3 4 5

16 17 18 19 6

15 24 25 20 7

14 23 22 21 8

13 12 11 10 9

Total power point:3

(0,0)

(4,2)

(3,2)

Solution:**C:**

#include <stdio.h>

int main()

{

int a[10][10],n,i,j,d=0,c=1,count=0;

scanf("%d",&n);

for(i=0;i<10;i++)

for(j=0;j<10;j++)

{

if(i>=n || j>=n)

a[i][j]=-1;

else

```
        a[i][j]=0;
    }
    for(c=1,i=0,j=0;c<=n*n;c++)
    {
        if(a[i][j]==0 && d==0)
        {
            a[i][j]=c;

            j++;
        }
        else if(a[i][j]==0 && d==1)
        {
            a[i][j]=c;

            i++;
        }
        else if(a[i][j]==0 && d==2)
        {
            a[i][j]=c;

            j--;
        }
        else if(a[i][j]==0 && d==3)
        {
            a[i][j]=c;
```

```
        i--;  
    }  
    if(a[i][j]!=0)  
    {  
        if(d==0)  
        {  
            i=i+1;  
            j=j-1;  
            d=1;  
        }  
        else if(d==2)  
        {  
            i=i-1;  
            j=j+1;  
            d=3;  
        }  
        else if(d==1)  
        {  
            i=i-1;  
            j=j-1;  
            d=2;  
        }  
    }
```

```
        else if(d==3)
        {
            i=i+1;

            j=j+1;

            d=0;

        }
    }

    for(i=0;i<n;i++)
    {

        printf("\n");

        for(j=0;j<n;j++)

            printf("%d ",a[i][j]);

    }

    for(i=0;i<n;i++)

        for(j=0;j<n;j++)

            if(a[i][j]%11==0)

                count+=1;

    printf("Total Power points :%d\n",count+1);

    printf("(0,0)");
```

```
for(i=0;i<n;i++)
    for(j=0;j<n;j++)
        if (a[i][j] % 11 == 0)
            printf("(%d,%d)",i,j);
return 0;
}
```

Java:

```
package bitManipulation;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.*;

class Main
{
    public static void main(String args[])throws IOException
    {
        BufferedReader br=new BufferedReader(new
        InputStreamReader(System.in));

        int no=Integer.parseInt(br.readLine());

        int A[][]=new int[no][no];
```

```
int x[]=new int[no];

int y[]=new int[no];

int k=1,a=1,col1=0, col2=no-1, row1=0, row2=no-1;

while(k<=no*no)

{

    for(int i=col1;i<=col2;i++)

    {

        A[row1][i]=k++;

        if(A[row1][i]%11==0)

        {

            x[a]=row1;

            y[a]=i;

            a++;

        }

    }

    for(int j=row1+1;j<=row2;j++)

    {

        A[j][col2]=k++;

        if(A[j][col2]%11==0)
```

```
{  
    x[a]=j;  
    y[a]=col2;  
    a++;  
}  
}
```

```
for(int i=col2-1;i>=col1;i--)  
{  
    A[row2][i]=k++;  
    if(A[row2][i]%11==0)  
    {  
        x[a]=row2;  
        y[a]=i;  
        a++;  
    }  
}
```

```
for(int j=row2-1;j>=row1+1;j--)  
{  
    A[j][col1]=k++;  
    if(A[j][col1]%11==0)
```



```

        {
            x[a]=j;
            y[a]=col1;
            a++;
        }
    }

    col1++;
    col2--;
    row1++;
    row2--;
}

for(int i=0;i<no;i++)
{
    for(int j=0;j<no;j++)
    {
        System.out.print(A[i][j]+" ");
    }

    System.out.print("\n");
}

System.out.print("Total Power points : "+a+"\n(0,0) \n");

for(int i=1;i<a;i++)

```

```

    {
        System.out.print("(" + x[i] + ", " + y[i] + ") \n");
    }
}
}

```

Python:

```
n = int (input ("Enter the size of matrix:"))
```

```
r = n
```

```
c = n
```

```
t = 1
```

```
x = 0
```

```
y = 0
```

```
count=0
```

```
dict={}
```

```
a = [[0 for i in range (n)] for j in range (n)]
```

```
while (x <= r and y <= c):
```

```
    for i in range (y, c, +1):
```

```
        a[y][i] = t
```

```
        t += 1
```

```
    x += 1
```

```
for i in range (x, r, +1):
```

```
    a[i][c - 1] = t
```

```
    t += 1
```

```
c -= 1
```

```
for i in range (c, y, -1):
```

```
    a[c][i - 1] = t
```

```
    t += 1
```

```
r -= 1
```

```
for i in range (r, x, -1):
```

```
    a[i - 1][y] = t
```

```
    t += 1
```

```
y += 1
```

```
for i in range (0, n):
```

```
    for j in range (0, n):
```

```
        print ((a[i][j]),'\t', end=" ")
```

```
    print ()
```

```
for i in range (0, n):
```

```

for j in range (0, n):
    if(a[i][j]%11==0):
        count+=1
print('Total Power points :',count+1)

```

```

print('(0,0)')
for i in range (0, n):
    for j in range (0, n):
        if (a[i][j] % 11 == 0):
            dict[a[i][j]] = (i,j)
            print('(',i,',',j,')', sep='')

```

3. The casino has introduced a new game in which there are M vertical chutes each containing N single digit (possibly zero) numbers. You can choose any chute and draw the bottom number and when you do this all the other numbers in the chute descend by one slot. You need to build the largest integer using this process drawing all the numbers from the chutes. For example, in the following example, we have three chutes of four numbers each and the largest number that can be drawn is 469534692781. Given the number of chutes and the numbers in each chute, write a program to find the largest integer that could be formed using the above process. To find the largest integer that could be formed using the above process.

9	1	5
6	8	9
4	7	6
3	2	4

Constraints

$1 \leq M \leq 20$, $1 \leq N \leq 50$

Input Format

First line contains M,N two comma separated integers giving the number of chutes and the number of digits in each chute

The next M lines each contain N comma separated digits, giving the digits from top to bottom in each of the chutes.

Output

One line containing the largest integer that could be formed.

Example 1

Input

2,3

1,2,3

2,4,6

1	2
2	4
3	6

Output

643221

Explanation

M is 2 and N is 3 (there are 2 chutes with 3 digits in each). The chutes look like this

The largest integer that can be formed is 643221

Example 2**Input**

4,4

7,5,5,2

3,6,1,7

8,7,0,4

8,7,3,9

7	3	8	8
5	6	7	7
5	1	0	3
2	7	4	9

Output

9743782557163078

Explanation

M is 4 and N is 4. The chutes look like this

The largest integer that can be formed is 9743782557163078, and this is the output.

Solution:**C:**

```
#include<stdio.h>
```

```
int main()
```

```
{
```

```
    int a[20][20],i,j,m,n,f=0,c=0,r,p,k;
```

```
    char ch;
```

```
    scanf("%d%c%d",&m,&ch,&n);
```

```
    for(i=0;i<m;i++)
```

```
    {
```

```
        for(j=0;j<n;j++)
```

```
        {
```

```
            scanf("%d",&a[i][j]);
```

```
        if(j<n-1)
            scanf("%c",&ch);

    }
}

while(f!=1)
{
    j=n-1;c=-1;
    for(k=m-1;k>=0;k--)
    {
        if(a[k][j]>c)
        {
            c=a[k][j];
            r=k;

        }

    }

    for(p=j;p>0;p--)
    {
        a[r][p]=a[r][p-1];
```

```
}  
  
a[r][p]=-1;  
  
if(c!=-1)  
  
printf("%d",c);  
  
else  
  
break;  
  
}  
  
}
```

Java:

```
package bitManipulation;  
  
import java.util.ArrayList;  
  
import java.util.Scanner;  
  
  
public class Main {  
  
  
  
    public static void main(String[] args) {  
  
        Scanner sc = new Scanner(System.in);  
  
        int[][] a=new int[100][100];  
  
        int i,j,m,n,f=0,c=0,r=0,p,k;  
  
        char ch;  
  
        m=sc.nextInt();  
  
        ch=sc.next().charAt(0);
```



```
n=sc.nextInt();  
for(i=0;i<m;i++)  
{  
    for(j=0;j<n;j++)  
    {  
        a[i][j]=sc.nextInt();  
        if(j<n-1)  
            ch=sc.next().charAt(0);  
    }  
}  
  
while(f!=1)  
{  
    j=n-1;c=-1;  
    for(k=m-1;k>=0;k--)  
    {  
        if(a[k][j]>c)  
        {  
            c=a[k][j];  
            r=k;  
        }  
    }  
}
```

```

    }

    }
    for(p=j;p>0;p--)
    {
        a[r][p]=a[r][p-1];
    }
    a[r][p]=-1;
    if(c!=-1)
        System.out.printf("%d",c);
    else
        break;
}

```

```

    }
}

```

Python:

```
m, n = map(int, input().strip().split(","))
```

```
a = []
```

```

for i in range(m):
    a.append(list(map(int, input().strip().split(","))))

high = [0]*m

for i in range(m):
    high[i] = a[i][-1]

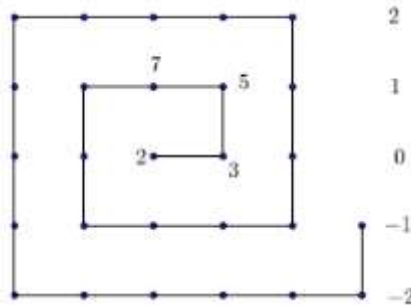
ans = []

for i in range(m*n):
    h = max(high)
    ans.append(str(h))
    if len(a[high.index(h)]) != 0:
        a[high.index(h)].pop()
    if len(a[high.index(h)]) == 0:
        high[high.index(h)] = -1
    else:
        high[high.index(h)] = a[high.index(h)][-1]

print("".join(ans))

```

4. The prime numbers are written in a spiral form starting at (0,0) and moving as shown in the diagram below. The numbers shown on the right column and the bottom row are the column numbers and row numbers respectively (y and x coordinate frames).



The objective is to find the position (x and y coordinates) of a given prime.

Input Format:

The input consists of multiple lines.

The first line gives the number of primes (N) in this test case.

The next N lines contain one prime in each line.

Output Format:

The output consists of N lines.

Each consists of a space separated pair of integers giving the x and y coordinates of the corresponding prime in the input.

Constraints:

$N \leq 10$

Each prime < 1000000

Example 1

Input

2
3
7

Output

1 0
0 1

Explanation

There are 2 primes in this test case ($N=2$). The primes are 3 and 7.

The coordinates of these in the spiral is (1,0) and (0,1).

The output hence has these in space separated form.

Example 2

Input

3
5
11
13

Output

1 1
1
1
1
0

Explanation

There are 3 primes in this test case (N=2). The primes are 5, 11 and 13. The coordinates of these in the spiral is (1,1), (1,1) and (1,0). The output hence has these in space separated form.

Solution:

C:

```
#include<stdio.h>

int prime[100];

int a[9][10],num,search[100];

void create()
{
    int i=4,j=4,n=3,m=5,n1=5,m1=3,k=0,dire=0;
    while(i!=-1)
    {
        a[i][j]=prime[k];

        k++;

        if(dire==0)
```

```
{  
    j++;  
    if(j==m)  
    {  
        m++;  
        dire=1;  
    }  
}  
else if(dire==1)  
{  
    i--;  
    if(i==n)  
    {  
        n--;  
        dire=2;  
    }  
}  
else if(dire==2)  
{  
    j--;  
    if(j==m1)  
    {
```

```
        m1--;

        dire=3;

    }

}

else if(dire==3)

{

    i++;

    if(i==n1)

    {

        n1++;

        dire=0;

    }

}

}

for(k=0;k<num;k++)

{

    for(i=0;i<9;i++)

    {

        for(j=0;j<10;j++)

        {

            if(a[i][j]==search[k])

            {
```

```
        printf("%d %d\n",j-4,4-i); //printing the position as required
    }
}
}
}

}

int main()
{
    int n, i = 3, count, c,j=0;
    n=100;
    //Reading Input
    scanf("%d",&num);
    for(j=0;j<num;j++)
        scanf("%d",&search[j]);
    j=0;
    if ( n >= 1 )
    {
        prime[j]=2;
        j++;
    }

    for ( count = 2 ; count <= n ; )
```



```
{  
    for ( c = 2 ; c <= i - 1 ; c++ )  
    {  
        if ( i%c == 0 )  
            break;  
    }  
    if ( c == i )  
    {  
        prime[j]=i;  
        j++;  
        count++;  
    }  
    i++;  
}  
create();  
return 0;  
}
```

Java:

```
import java.util.*;  
  
public class Main {  
  
    static void spiralSplicer(int input)
```

```
{  
    Scanner ip=new Scanner(System.in);  
    int step_count = ip.nextInt();  
    int step_limit = ip.nextInt();  
    int adder = ip.nextInt();  
    int x = 0, y = 0;  
    for (int n = 2; n != input + 1; n++, step_count++)  
    {  
        if (step_count <= .5 * step_limit) {  
            x += adder;  
            System.out.print(x); }  
        else if (step_count <= step_limit)  
            y += adder;  
  
        if (step_count == step_limit)  
        {  
            adder *= -1;  
            step_limit += 2;  
            step_count = 0;  
        }  
    }  
    System.out.print(y);
```

```
}  
  
static int primeIndex(int input)  
{  
  
    int j, cnt, prime_c = 0;  
    for (int i = 2; i <= 1000000; i++)  
    {  
  
        cnt = 0;  
        for (j = 2; j <= i; j++)  
        {  
  
            if (i % j == 0)  
                cnt++;  
  
        }  
        if (cnt == 1)  
        {  
  
            prime_c++;  
            if (input == i)  
            {  
  
                input = prime_c;  
                break;  
  
            }  
  
        }  
  
    }  
  
}
```

```

        return input;
    }

    public static void main(String args[]) {
        Scanner ip=new Scanner(System.in);
        int input = ip.nextInt();
        spiralSplicer(primeIndex(input));
    }
}

```

Python:

```

def spiralSplicer(inp):
    step_count = int(input())
    step_limit = int(input())
    adder = int(input())
    x, y = 0, 0
    for n in range(2, inp + 1):

        if (step_count <= .5 * step_limit):
            x += adder

            print(x)

        elif (step_count <= step_limit):

```

```

    y += adder

if (step_count == step_limit):
    adder *= -1
    step_limit += 2
    step_count = 0
    step_count += 1

print (y )

def primeIndex(inp):
    cnt, prime_c = 0, 0
    for i in range(2, 1000000 + 1):
        cnt = 0
        for j in range(2, i + 1):
            if (i % j == 0):
                cnt += 1

        if (cnt == 1):
            prime_c += 1
            if (inp == i):
                inp = prime_c
                break

    return inp

inp = int(input())

temp = primeIndex(inp)

```

spiralSplicer(temp)

5. In a social network, online communities refer to the group of people with an interest towards the same topic. People connect with each other in a social network. A connection between Person I and Person J is represented as C I J. When two persons belonging to different communities connect, the net effect is merger of both communities which I and J belonged to.

We are only interested in finding out the communities with the member count being an even number. Your task is to find out those communities.

Input Format:

Input will consist of three parts, viz.

1. The total number of people on the social network (N)

2. Queries

- C I J, connect I and J

- Q 0 0, print the number of communities with even member-count -

1 will represent end of input.

Output Format:

For each query Q, output the number of communities with even member-count

Constraints:

$1 \leq N \leq 10^6$

$1 \leq I, J \leq N$

Test Cases

Input

5

Q 0 0

C 1 2

Q 0 0

C 2 3

Q 0 0

C 4 5

Q 0 0

-1

OUTPUT

0

1

0

1

Explanation:

For first query there are no members in any of the groups hence answer is 0.

After C 1 2, there is a group (let's take it as G1) with 1 and 2 as members hence total count at this moment is 1.

After C 2 3 total members in G1 will become {1, 2, 3} hence there are no groups with even count.

After C 4 5 there formed a new group G2 with {4, 5} as members, hence the total groups with even count is 1.

Solution:**C++:**

```
#include<stdio.h>

int comm[1000001],n;

void mergecom(int,int);

//void print();

int even(int);

int main(){

    int x,y,comNo=1;

    char c;

    while(1){
```

```
scanf("%c",&c);  
if(c=='C'){  
    scanf("%d %d",&x,&y);  
    if(comm[x]==0 && comm[y]==0){  
        comm[x]=comm[y]=comNo;  
        comNo++;  
    }  
    else if(comm[x]==0){  
        comm[x]=comm[y];  
    }  
    else if(comm[y]==0){  
        comm[y]=comm[x];  
    }  
    else if(comm[x]>comm[y]){  
        mergecom(comm[y],comm[x]);  
        comNo--;  
    }  
    else if(comm[y]>comm[x]){  
        mergecom(comm[x],comm[y]);  
        comNo--;  
    }  
}
```



```
else if(c=='Q'){
    scanf("%d %d",&x,&y);
    printf("%d",even(comNo));
}
else
    break;
}
scanf("%d",&n);
return 0;
}
```

```
void mergecom(int x,int y){
    for(int i=0;i<=n;i++){
        if(comm[i]==y)
            comm[i]=x;
        else if(comm[i]>y)
            comm[i]--;
    }
}
```

```
}
int even(int comNo){
    int count =0,temp=0;
    for(int i=1;i<comNo;i++){
```

```

for(int j=0;j<=n;j++){
    if(comm[j]==i)
        temp++;
}
if(temp%2==0 && temp!=0){
    count++;
}
temp=0;
}
return count;
}
/*
void print(){
for(int i=0;i<=n;i++)
    cout<<i<<comm[i]<<" ";
cout<<endl;
}*/

```

Java:

```

package bitManipulation;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

```

```
public class Main {  
  
    public static void main(String[] args) throws IOException {  
  
        BufferedReader bufferedReader = new  
        BufferedReader(new InputStreamReader(System.in));  
  
        int N = Integer.parseInt(bufferedReader.readLine());  
  
        int[] users = new int[N + 1],  
            groupSizes = new int[N / 2 + 1];  
  
        int group = 0;  
  
        String line;  
        while (!(line = bufferedReader.readLine()).equals("-1")) {  
            if (line.startsWith("C")) {  
                String[] inputs = line.split(" ");  
                int I = Integer.parseInt(inputs[1]),  
                    J = Integer.parseInt(inputs[2]);  
  
                if (users[I] != 0) {  
                    users[J] = users[I];  
                }  
            }  
        }  
    }  
}
```

```

        groupSizes[users[I]]++;
    } else if (users[J] != 0) {
        users[I] = users[J];
        groupSizes[users[J]]++;
    } else {
        group++;
        users[I] = group;
        users[J] = group;
        groupSizes[group] = 2;
    }
} else if (line.equals("Q 0 0")) {
    if (group == 0) {
        System.out.println("0");
        continue;
    }

    int evenGroups = 0;
    for (int groupSize : groupSizes) {
        if (groupSize > 0 && groupSize % 2 == 0) {
            evenGroups++;
        }
    }
}

```

```

    System.out.println(evenGroups);
}
}
}
}
}

```

6. Given a square maze (A) of dimension N, every entry (A_{ij}) in the maze is either an open cell 'O' or a wall 'X'. A rat can travel to its adjacent locations (left, right, top and bottom), but to reach a cell, it must be open. Given the locations of R rats, can you find out whether all the rats can reach others or not?

Input Format:

Input will consist of three parts, viz.

1. Size of the maze (N)
2. The maze itself ($A = N * N$)
3. Number of rats (R)
4. Location of R rats (X_i, Y_i)

Note:

(X_i, Y_i) will represents the location of the i-th rat.

Locations are 1-index based.

Output Format:

Print "Yes" if the rats can reach each other, else print "No"

Constraints:

 $1 \leq N \leq 350$
$$A_{ij} = \{'O', 'X'\}$$
$$1 \leq R \leq N * N$$
$$1 \leq X_i \leq N$$
$$1 \leq Y_i \leq N$$

Input

3
 O O X
 O X O
 O O X
 4
 1 1
 1 2
 2 1
 3 2

Output : Yes

Input

3
 O O X
 O X O
 O O X
 4
 1 1
 1 2
 2 1
 2 3

Output : No

Solution:

C++:

```
#include <iostream>
```

```
#include <string>
```

```
using namespace std;
```

```
const int MAX_N = 350;
```

```
// depth-first-search algorithm
```

// takes the maze, the size of the maze. the current position in the maze

```
int dfs(string maze[], int N, int x, int y) {  
    int rats = 0;  
  
    if (x < 0 || x >= N) return rats; // the position is not in the maze  
  
    if (y < 0 || y >= N) return rats; // the position is not in the maze  
  
    if (maze[x][y] != 'O' && maze[x][y] != 'R') return rats; // the position  
    isn't open  
  
    if (maze[x][y] == 'R') ++rats;  
  
    maze[x][y] = 'X'; // make the cell closed beacuse we already visited  
    it  
  
    static const int dx[4] = {-1, 0, 1, 0};  
    static const int dy[4] = {0, 1, 0, -1};  
  
    // visit all adjacent cells  
    for (int i = 0; i < 4; ++i) {  
        int nx = x + dx[i];  
        int ny = y + dy[i];  
        rats += dfs(maze, N, nx, ny);  
    }  
  
    return rats;  
}  
  
int main() {  
  
    int N;
```

```
cin >> N;

string maze[MAX_N];

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

    cin >> maze[i];

}

int R;

cin >> R;

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

    int X, Y;

    cin >> X >> Y;

    --X; --Y;

    maze[X][Y] = 'R';

}

// run the depth-first-search algorithm for unvisited cell

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

    for (int j = 0; j < N; ++j) {

        int rats = dfs(maze, N, i, j);

        if (rats == R) { // all rats are in the same connected component

            cout << "Yes" << endl;

            return 0;

        } else if (rats > 0) { // there is a rat in a component with not all
other rats
```



```

        cout << "No" << endl;

        return 0;
    }

}

}

}

```

Java:

```

package bitManipulation;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class Main {

    public static void main(String[] args) throws IOException {

        BufferedReader bufferedReader = new
        BufferedReader(new InputStreamReader(System.in));
    }
}

```

```
int N = Integer.parseInt(bufferedReader.readLine());
```

```
char[][] A = new char[N][N];
```

```
for (int i = 0; i < N; i++) {
```

```
    A[i] = bufferedReader.readLine().toCharArray();
```

```
}
```

```
int R = Integer.parseInt(bufferedReader.readLine());
```

```
int[] X = new int[R],
```

```
    Y = new int[R];
```

```
for (int i = 0; i < R; i++) {
```

```
    String[] inputs = bufferedReader.readLine().split(" ");
```

```
    X[i] = Integer.parseInt(inputs[0]) - 1;
```

```
    Y[i] = Integer.parseInt(inputs[1]) - 1;
```

```
}
```

```
boolean[][] visited;
```

```
for (int i = 1; i < R; i++) {
```

```
    visited = new boolean[N][N];
```

```
    if (!visit(A, X[i], Y[i], X[0], Y[0], visited)) {
```

```

        System.out.println("No");

        return;
    }
}

```

```

        System.out.println("Yes");
    }
}

```

```

private static boolean visit(char[][] A, int X1, int Y1, int X2, int
Y2, boolean[][] visited) {

```

```

    if (X1 == X2 && Y1 == Y2) {
        return true;
    }

```

```

    if (A[X1][Y1] == 'X' || visited[X1][Y1]) {
        return false;
    }

```

```

    visited[X1][Y1] = true;

```

```

    if (X1 - 1 >= 0 && visit(A, X1 - 1, Y1, X2, Y2, visited)) {
        return true;
    }
}

```

```

    }

    if (X1 + 1 < A.length && visit(A, X1 + 1, Y1, X2, Y2, visited))
    {
        return true;
    }

    if (Y1 - 1 >= 0 && visit(A, X1, Y1 - 1, X2, Y2, visited)) {
        return true;
    }

    if (Y1 + 1 < A.length && visit(A, X1, Y1 + 1, X2, Y2, visited))
    {
        return true;
    }

    return false;
}
}

```

7. You are given a square matrix of dimension N. Let this matrix be called A. Your task is to rotate A in clockwise direction by S degrees,

where S is angle of rotation. On the matrix, there will be 3 types of operations viz.

1. Rotation

Rotate the matrix A by angle S , presented as input in form of $A S$

2. Querying

Query the element at row K and column L , presented as input in form of $Q K L$

3. Updation

Update the element at row X and column Y with value Z , presented as input in form of $U X Y Z$

Print the output of individual operations as depicted in Output Specification

Input Format:

Input will consist of three parts, viz.

1. Size of the matrix (N)
 2. The matrix itself ($A = N * N$)
 3. Various operations on the matrix, one operation on each line.
(Beginning either with A , Q or U)
- 1 will represent end of input.

Note:

Angle of rotation will always be multiples of 90 degrees only.

All Update operations happen only on the initial matrix. After update all the previous rotations have to be applied on the updated matrix

Output Format:

For each Query operation print the element present at K - L location of the matrix in its current state.

Constraints:

$$1 \leq N \leq 1000$$

$$1 \leq A_{ij} \leq 1000$$

$$0 \leq S \leq 160000$$

$$1 \leq K, L \leq N$$

$$1 \leq Q \leq 100000$$

Test Cases

Input

2

1 2

3 4

A 90

Q 1 1

Q 1 2

A 90

Q 1 1

U 1 1 6

Q 2 2

-1

Output

3

1

4

6

Explanation:

Initial Matrix

1 2

3 4

After 90 degree rotation, the matrix will become

3 1

4 2

Now the element at A11 is 3 and A12 is 1.

Again the angle of rotation is 90 degree, now after the rotation the matrix will become

4 3

2 1

Now the element at A11 is 4.

As the next operation is **Update**, update initial matrix i.e.

6 2

3 4

After updating, apply all the previous rotations (i.e. 180 = two 90 degree rotations).

The matrix will now become

4 3

2 6

Now A22 is 6.

Solution:

C:

```
#include<stdio.h>
```

```
int main() {
```

```
    /* Enter your code here. Read input from STDIN. Print output to  
    STDOUT */
```

```
    int n;
```

```
    scanf("%d",&n);
```

```
    int a[n][n],i,j,s=0;
```

```
    for(i=0;i<n;i++)
```

```
    {
```

```
        for(j=0;j<n;j++)
```

```
            scanf("%d",&a[i][j]);
```

```
    }
```

```
    while(1)
```

```
    {
```

```
        char c;
```

```
        scanf(" %c",&c);
```

```
        if(c == '-')
```

```
    return 0;

else if(c=='A')
{
    int x;

    scanf("%d",&x);

    s=s+x/90;

    s=s%4;
}

else if(c == 'Q')
{
    int x,y;

    scanf("%d%d",&x,&y);

    x--;

    y--;

    if(s == 0) printf("%d\n",a[x][y]);

    if(s== 1) printf("%d\n",a[n-y-1][x]);

    if(s== 2) printf("%d\n",a[n-x-1][n-y-1]);

    if(s== 3) printf("%d\n",a[y][n-x-1]);

}

else if(c == 'U')
{

    int x,y,z;
```



```
scanf("%d%d%d",&x,&y,&z);

a[x-1][y-1]=z;

}

}

return 0;

}
```

Java:

```
package bitManipulation;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class Main {

    public static void main(String[] args) throws IOException {

        BufferedReader bufferedReader = new
        BufferedReader(new InputStreamReader(System.in));

        int N = Integer.parseInt(bufferedReader.readLine());

        int[][] A = new int[N][N];
```

```
for (int i = 0; i < N; i++) {  
    String[] inputs = bufferedReader.readLine().split(" ");  
    for (int j = 0; j < N; j++) {  
        A[i][j] = Integer.parseInt(inputs[j]);  
    }  
}
```

```
int angle = 0;
```

```
int[][] RA = new int[N][N];  
copy(A, RA, N);
```

```
String line;  
while (!(line = bufferedReader.readLine()).equals("-1")) {  
    if (line.startsWith("A")) {  
        int S = Integer.parseInt(line.split(" ")[1]);  
        angle += S;  
  
        rotate(RA, N, S / 90);  
    } else if (line.startsWith("Q")) {  
        String[] inputs = line.split(" ");  
        int K = Integer.parseInt(inputs[1]),
```

```

        L = Integer.parseInt(inputs[2]);

        System.out.println(RA[K - 1][L - 1]);
    } else if (line.startsWith("U")) {
        String[] inputs = line.split(" ");
        int X = Integer.parseInt(inputs[1]),
            Y = Integer.parseInt(inputs[2]),
            Z = Integer.parseInt(inputs[3]);

        A[X - 1][Y - 1] = Z;

        RA = new int[N][N];
        copy(A, RA, N);

        rotate(RA, N, angle / 90);
    }
}
}

```

```

private static void copy(int[][] A, int[][] RA, int N) {
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {

```

```
        RA[i][j] = A[i][j];
    }
}
}
```

```
private static void rotate(int[][] RA, int N) {
    int[][] temp = new int[N][N];
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            temp[i][j] = RA[N - j - 1][i];
        }
    }

    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            RA[i][j] = temp[i][j];
        }
    }
}
```

```
private static void rotate(int[][] RA, int N, int rotations) {
    for (int i = 1; i <= rotations; i++) {
```

```

        rotate(RA, N);
    }
}

}

```

Python:

```

def rotate90Clockwise(A):
    N = len(A[0])
    for i in range(N // 2):
        for j in range(i, N - i - 1):
            temp = A[i][j]
            A[i][j] = A[N - 1 - j][i]
            A[N - 1 - j][i] = A[N - 1 - i][N - 1 - j]
            A[N - 1 - i][N - 1 - j] = A[j][N - 1 - i]
            A[j][N - 1 - i] = temp
    return A

n = int(input())
original_mat = []
for i in range(n):
    l = list(map(int, input().strip().split()))
    original_mat.append(l)

```

```
copy_mat = list(map(list, original_mat))

angle_count = 0

while 1:

    query = input().split()

    if query[0] == 'A':

        count = int(query[1])

        count = count // 90

        count = count % 4

        angle_count += count

        for _ in range(count):

            original_mat = rotate90Clockwise(original_mat)

    elif query[0] == 'Q':

        r , c = int(query[1]) , int(query[2])

        print(original_mat[r-1][c-1])

    elif query[0] == 'U':

        r , c , val = int(query[1]) , int(query[2]) , int(query[3])

        duplicate_matrix = list(map(list,copy_mat))

        duplicate_matrix[r-1][c-1] = val

        for _ in range(angle_count % 4):

            original_mat = rotate90Clockwise(duplicate_matrix)

    elif query[0] == '-1':

        exit()
```

8. Given an $M \times N$ matrix, with a few hurdles arbitrarily placed, calculate the cost of longest possible route from point A to point B within the matrix.

Input Format:

1. First line contains 2 numbers delimited by whitespace where, first number M is number of rows and second number N is number of columns
2. Second line contains number of hurdles H followed by H lines, each line will contain one hurdle point in the matrix.
3. Next line will contain point A, starting point in the matrix.
4. Next line will contain point B, stop point in the matrix.

Output Format:

Output should display the length of the longest route from point A to point B in the matrix.

Constraints:

1. The cost from one position to another will be 1 unit.
2. A location once visited in a particular path cannot be visited again.
3. A route will only consider adjacent hops. The route cannot consist of diagonal hops.
4. The position with a hurdle cannot be visited.
5. The values $M \times N$ signifies that the matrix consists of rows ranging from 0 to $M-1$ and columns ranging from 0 to $N-1$.
6. If the destination is not reachable or source/ destination overlap with hurdles, print cost as -1.

Example 1

Input

```
1 0 1 1 1 1 0 1 1 1
1 0 1 0 1 1 1 0 1 1
1 1 1 0 1 1 0 1 0 1
0 0 0 0 1 0 0 1 0 0
1 0 0 0 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 0
1 0 0 0 1 0 0 1 0 1
```

```

1 0 1 1 1 1 0 0 1 1
1 1 0 0 1 0 0 0 0 1
1 0 1 1 1 1 0 1 0 0
0 0
5 7

```

Output

24

EXPLANATION

Here matrix will be of size 3x10 matrix with a hurdle at (1,2),(1,5) and (1,8) with starting point A(0,0) and stop point B(1,7)

```

3 10
3 -- (no. of hurdles )
1 2
1 5
1 8
0 0 -- (position of A)
1 7 -- (position of B)

```

So if you examine matrix below shown in Fig 1, total hops

(->) count is 24. So final answer will be 24. No other route longer than this one is possible in this matrix.

Solution:

C:

```
#include <stdio.h>
```

```
#define M 10
```

```
#define N 10
```

```
bool isSafe(int mat[M][N], int visited[M][N], int x, int y)
```

```
{
```



```

        if (mat[x][y] == 0 || visited[x][y])
            return false;

        return true;
    }

    bool isValid(int x, int y)
    {
        if (x < M && y < N && x >= 0 && y >= 0)
            return true;

        return false;
    }

    void findLongestPath(int mat[M][N], int visited[M][N], int i, int j,
                        int x, int y, int& max_dist, int dist)
    {
        if (mat[i][j] == 0)
            return;

        if (i == x && j == y)
        {
            max_dist = max(dist, max_dist);
            return;
        }
    }

```

```

visited[i][j] = 1;

if (isValid(i + 1, j) && isSafe(mat, visited, i + 1, j))
    findLongestPath(mat, visited, i + 1, j, x, y, max_dist, dist +
1);

if (isValid(i, j + 1) && isSafe(mat, visited, i, j + 1))
    findLongestPath(mat, visited, i, j + 1, x, y, max_dist, dist +
1);

if (isValid(i - 1, j) && isSafe(mat, visited, i - 1, j))
    findLongestPath(mat, visited, i - 1, j, x, y, max_dist, dist +
1);

if (isValid(i, j - 1) && isSafe(mat, visited, i, j - 1))
    findLongestPath(mat, visited, i, j - 1, x, y, max_dist, dist +
1);

visited[i][j] = 0;
}

```

```

int main()
{
    int mat[M][N] =
    {
        { 1, 0, 1, 1, 1, 1, 0, 1, 1, 1 },
        { 1, 0, 1, 0, 1, 1, 1, 0, 1, 1 },
        { 1, 1, 1, 0, 1, 1, 0, 1, 0, 1 },

```

```

        { 0, 0, 0, 0, 1, 0, 0, 1, 0, 0 },
        { 1, 0, 0, 0, 1, 1, 1, 1, 1, 1 },
        { 1, 1, 1, 1, 1, 1, 1, 1, 1, 0 },
        { 1, 0, 0, 0, 1, 0, 0, 1, 0, 1 },
        { 1, 0, 1, 1, 1, 1, 0, 0, 1, 1 },
        { 1, 1, 0, 0, 1, 0, 0, 0, 0, 1 },
        { 1, 0, 1, 1, 1, 1, 0, 1, 0, 0 }

};

int visited[M][N];

memset(visited, 0, sizeof visited);

int max_dist = 0;

findLongestPath(mat, visited, 0, 0, 5, 7, max_dist, 0);

printf("%d", max_dist);

return 0;

}

```

Java:

```

class Main

{

    private static final int M = 10;

    private static final int N = 10;

    private static boolean isSafe(int mat[][], int visited[][], int x, int
y)

```

```

{
    if (mat[x][y] == 0 || visited[x][y] != 0)
        return false;

    return true;
}

private static boolean isValid(int x, int y)
{
    if (x < M && y < N && x >= 0 && y >= 0)
        return true;

    return false;
}

public static int findLongestPath(int mat[][], int visited[][], int i,
                                int j, int x, int y, int
max_dist, int dist)
{
    if (mat[i][j] == 0) {
        return 0;
    }

    if (i == x && j == y)
    {

        return Integer.max(dist, max_dist);
    }
}

```

```

    }

    visited[i][j] = 1;

    if (isValid(i + 1, j) && isSafe(mat, visited, i + 1, j)) {
        max_dist = findLongestPath(mat, visited, i + 1, j, x, y,
                                   max_dist, dist
+ 1);
    }

    if (isValid(i, j + 1) && isSafe(mat, visited, i, j + 1)) {
        max_dist = findLongestPath(mat, visited, i, j + 1, x, y,
                                   max_dist, dist
+ 1);
    }

    if (isValid(i - 1, j) && isSafe(mat, visited, i - 1, j)) {
        max_dist = findLongestPath(mat, visited, i - 1, j, x, y,
                                   max_dist, dist
+ 1);
    }

    if (isValid(i, j - 1) && isSafe(mat, visited, i, j - 1)) {
        max_dist = findLongestPath(mat, visited, i, j - 1, x, y,
                                   max_dist, dist
+ 1);
    }

    visited[i][j] = 0;

```

```
        return max_dist;
    }

    public static void main(String[] args)
    {
        int mat[][] =
        {
            { 1, 0, 1, 1, 1, 1, 0, 1, 1, 1 },
            { 1, 0, 1, 0, 1, 1, 1, 0, 1, 1 },
            { 1, 1, 1, 0, 1, 1, 0, 1, 0, 1 },
            { 0, 0, 0, 0, 1, 0, 0, 1, 0, 0 },
            { 1, 0, 0, 0, 1, 1, 1, 1, 1, 1 },
            { 1, 1, 1, 1, 1, 1, 1, 1, 1, 0 },
            { 1, 0, 0, 0, 1, 0, 0, 1, 0, 1 },
            { 1, 0, 1, 1, 1, 1, 0, 0, 1, 1 },
            { 1, 1, 0, 0, 1, 0, 0, 0, 0, 1 },
            { 1, 0, 1, 1, 1, 1, 0, 1, 0, 0 }
        };

        int[][] visited= new int[N][N];

        int max_dist = findLongestPath(mat, visited, 0, 0, 5, 7, 0,
0);

        System.out.println("Maximum length path is " +
max_dist);
    }
}
```

```

    }
}

```

Python:

```

def isSafe(mat, visited, x, y):
    return not (mat[x][y] == 0 or visited[x][y])

def isValid(x, y):
    return M > x >= 0 and N > y >= 0

def findLongestPath(mat, visited, i, j, x, y, max_dist, dist):
    if mat[i][j] == 0:
        return 0

    if i == x and j == y:
        return max(dist, max_dist)

    visited[i][j] = 1

    if isValid(i + 1, j) and isSafe(mat, visited, i + 1, j):
        max_dist = findLongestPath(mat, visited, i + 1, j, x, y,
max_dist, dist + 1)

    if isValid(i, j + 1) and isSafe(mat, visited, i, j + 1):
        max_dist = findLongestPath(mat, visited, i, j + 1, x, y,
max_dist, dist + 1)

    if isValid(i - 1, j) and isSafe(mat, visited, i - 1, j):
        max_dist = findLongestPath(mat, visited, i - 1, j, x, y,
max_dist, dist + 1)

```

```

    if isValid(i, j - 1) and isSafe(mat, visited, i, j - 1):
        max_dist = findLongestPath(mat, visited, i, j - 1, x, y,
max_dist, dist + 1)
        visited[i][j] = 0
    return max_dist

if __name__ == '__main__':
    mat = [
        [1, 0, 1, 1, 1, 1, 0, 1, 1, 1],
        [1, 0, 1, 0, 1, 1, 1, 0, 1, 1],
        [1, 1, 1, 0, 1, 1, 0, 1, 0, 1],
        [0, 0, 0, 0, 1, 0, 0, 1, 0, 0],
        [1, 0, 0, 0, 1, 1, 1, 1, 1, 1],
        [1, 1, 1, 1, 1, 1, 1, 1, 1, 0],
        [1, 0, 0, 0, 1, 0, 0, 1, 0, 1],
        [1, 0, 1, 1, 1, 1, 0, 0, 1, 1],
        [1, 1, 0, 0, 1, 0, 0, 0, 0, 1],
        [1, 0, 1, 1, 1, 1, 0, 1, 0, 0]
    ]

```

M = N = 10

visited = [[0 for x in range(N)] for y in range(M)]

max_dist = findLongestPath(mat, visited, 0, 0, 5, 7, 0, 0)


```
print("Maximum length path is", max_dist)
```

SEARCHING AND SORTING

1. PROBLEM STATEMENT

We called it as Jumping Beetle

A beetle on a board of size $M \times M$ squares. Each square has coordinates, a pair of integers (i,j) , where, i is the row number and j is the column number. Associated with each square is the coordinates of the square it jumps to when it lands there. as an example, in the 6×6 board below, the beetle jumps to $(2,3)$ from $(1,1)$, and jumps to $(5,2)$ from $(6,4)$.

If the starting location is given, the target is to see the position of the beetle after a large NO. of jumps.

Input Format:

1st Line

Input has 3 comma separated positive integers M,A,B . M is the length of a side of the board (the board is of size $M \times M$). The NO. of jumps the beetle makes is AB (A multiplied by B)

The next M lines

consist of M pairs of comma separated positive integers, each pair separated by a semicolon(;). The M th pair in each line i.e k shows the coordinates of the square it jumps to if and only if it lands on square (k,m) .

Finally, there is one line with a comma separated pair of no. giving

the initial position of the Jumping Beetle

Output Format:

The output is the coordinates of the beetle's position after A,B moves

Constraints:

- $6 \leq M \leq 20$
- $1 \leq A, B \leq 109$

Example 1

Input:

6,2,3
 2,3;2,4;2,1;3,5;3,4;4,2
 4,2;4,1;3,1;3,6;4,4;1,4
 1,2;1,3;4,5;5,5;2,1;1,5
 6,2;6,1;2,2;5,6;2,6;2,5
 3,2;3,3;6,5;6,6;6,3;6,4
 5,3;5,4;5,1;5,2;4,6;1,6
 1,2

Output:

6,3

Example 2

Input:

6,12,2

2,3;2,4;2,1;3,5;3,4;4,2

4,2;4,1;3,1;3,6;4,4;1,4

1,2;1,3;4,5;5,5;2,1;1,5

6,2;6,1;2,2;5,6;2,6;2,5

3,2;3,3;6,5;6,6;6,3;6,4

5 ,3;5,4;5,1;5,2;4,6;1,6

4 ,3

Output:

6,1

C:

```
#include<stdio.h>
```

```
#include<string.h>
```

```
struct pair{
```

```
int first;
```

```
int second;
```

```
};
```

```
int main(){
```

```
int n,a,b;
```

```
char ch;
```

```
scanf("%d%c%d%c%d",&n,&ch,&a,&ch,&b);
```

```
struct pair A [n][n];
```

```
int dp[n][n];
```

```
memset(dp,-1,sizeof(dp));

for(int i=0;i<n;i++){
for(int j=0;j<n;j++){
if(j==n-1){
scanf("%d%c%d",&A[i][j].first,&ch,&A[i][j].second);
continue;
}
scanf("%d%c%d%c",&A[i][j].first,&ch,&A[i][j].second,&ch);
}
}

int inx,iny,fx,fy,q;

q=a*b; q++;

scanf("%d%c%d",&inx,&ch,&iny);

//printf("%d",iny);

fx=--inx;fy=--iny;

while(q--){
if(dp[fx][fy]!=-1){
int kx=A[fx][fy].first-1;
int ky=A[fx][fy].second-1;
fx=kx;
fy=ky;
dp[fx][fy]=q;
```

```
}  
else{  
    dp[fx][fy]=dp[fx][fy]-q;  
    q=q%dp[fx][fy];  
}  
}  
printf("%d,%d",fx+1,fy+1);  
}
```

2. PROBLEM STATEMENT

Given required text comprising of words & numbers, sort them both in increasing order & shown them in a manner that a word is followed by a number. Words can be in upper case or lower case. we need to convert them into lowercase as well as sort then and then print them.

Input Format:

First line: Total number of test cases N

Next N lines

each line contains a text in which words are at odd position & no. are at even position & are delimited by space

Output Format:

Words & numbers sorted in increasing order & printed in a manner

that a word is followed by a number.

Constraints:

1. Text starts with a word
2. Count of words & no. are the same.
3. Duplicate elements should not allowed
4. Words must be show in lower case.
5. In text No special characters allowed .

Sample Input and Output

Example 1

Input:

2

Sagar 35 sanjay 12 ganesh 53 ramesh 19

Ganesh 59 suresh 19 rakesh 26 lalit 96

Output:

ganesh 12 ramesh 19 sagar 35 sanjay 53

ganesh 19 lalit 26 rakesh 59 suresh 96

C:

```
#include<stdio.h>
```

```
#include<string.h>
```

```
#include<ctype.h>
```

```
int main()
```

```
{
```

```
char s[100][100],t[100],e[100][100],o[100][100];

int i=0,j,k=0,swap,p,n,c,d,m=0,q,w,a[1000];

do

{

    scanf("%s %d",&s[i],&a[i]);

    i++;

}

while(getchar() != '\n');

for(w=0;w<i;w++)

{

    q=0;

    while(s[w][q] != '\0')

    {

        s[w][q] = tolower(s[w][q]);

        q++;

    }

}

for(w=1;w<i;w++)

{

    for(j=1;j<i;j++)

    {

        if(strcmp(s[j-1],s[j])>0)
```



```
{  
    strcpy(t,s[j-1]);  
    strcpy(s[j-1],s[j]);  
    strcpy(s[j],t);  
}  
}  
}  
  
for (c = 0 ; c < i-1; c++)  
  
    {  
  
        for (d = 0 ; d < i - c - 1 ; d++)  
  
            {  
  
                if (a[d] > a[d+1])  
  
                {  
  
                    swap=a[d];  
  
                    a[d]=a[d+1];  
  
                    a[d+1]=swap;  
  
                }  
  
            }  
  
        }  
  
    }  
  
    for(w=0;w<i;w++)  
  
    {  
  
        printf("%s %d ",s[w],a[w]);
```

```

}

return 0;

}

```

Java:

```

import java.util.Arrays;

final class AlnumStringSort {

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

        final char[] input = args[0].toCharArray();

        Arrays.sort(input);

        final StringBuilder[] parts = {new StringBuilder(), new
        StringBuilder(), new StringBuilder(), new StringBuilder()};

        for (final char c : input) {

            if ('a' <= c && c <= 'z') {

                parts[0].append(c);

            } else if ('A' <= c && c <= 'Z') {

                parts[1].append(c);

            } else if ("02468".indexOf(c) != -1) {

                parts[2].append(c);

            } else {

                parts[3].append(c);

            }

        }

    }

}

```

```
final StringBuilder acc = parts[0];  
for (int i = 1; i < parts.length; i++) {  
    acc.append(parts[i]);  
}  
System.out.println(acc); }  
}
```

3. PROBLEM STATEMENT

Football League Table Statement

All major football leagues have huge league tables. Whenever a new match is played, the league table of match is updated to show the current rankings (based on Scores, Goals For (GF), Goals Against (GA)). Given the results of a few matches among teams, write a code to print all the names of the teams in increasing order based on their rankings.

Rules are as follow A win results in TWO points, a draw results in ONE point & a loss is worth ZERO points. The team with the most number of goals in a match wins the match. Goal Difference is calculated as Goals For GF: Goals Against GA. Football Teams can play a maximum of 2 matches against each other : Home & Away matches respectively.

The Football match ranking is decided as below :

Football Team with maximum points is ranked 1 & minimum points is placed last Ties are broken as follows Football Teams with same points are ranked according to Goal Difference

If Goal Difference(GD) is the equal , the Football team with higher Goals For is ranked ahead

If GF is equal, the teams must be at the equal rank but they must be printed in case-insensitive alphabetic according to the team names.

More than TWO matches of same teams, should be considered as Invalid I/P.

A team can't play matches against itself, Therefore if team names are same for a given match, it should be considered Invalid I/P .

Input Format:

Football teams [Na] delimited by a whitespace character

Third line: no. of matches M

Next M lines: a match information tuple {T1 T2 S1 S2}, where tuple is consisting of the following information

Name of the first team: T1

Name of the second team: T2

Goals scored by the first team: S1

Goals scored by the second team: S2

Output Format:

Team names in order of their rankings, one team / line

OR

where appropriate Print “Invalid Input”.

Constraints:

$0 < N \leq 10,000$ $0 \leq S1, S2$

Java:

```
import java.util.*;
import java.lang.*;

class Main
{
    public static void main(String args[]) throws NullPointerException
    {
        int n,m,i,j,k;
        int x=0;

        Scanner sc=new Scanner(System.in);

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

        System.out.println("Enter number of Teams:");
        n=sc.nextInt();

        String z[]=new String[n];

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

        System.out.println("Enter names of Teams:");

        for(i=0;i<n;i++)
        {
```

```

z[i]=sc.next();

}

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

System.out.println("Enter number of Matches:");

m=sc.nextInt();

int pt[]=new int[n];

int gf[]=new int[n];

int ga[]=new int[n];

int gd[]=new int[n];

int mt[]=new int[n];

String s[]=new String[n+n];

String a[][]=new String[m][4];

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

System.out.println("Enter details of Matches(Team1 Team2
T1goals T2goals)");

for(i=0;i<m;i++)

{

for(j=0;j<4;j++)

{

a[i][j]=sc.next();

}

}

System.out.println();

```

```
}
```

```
for(i=0;i<m;i++)
```

```
{
```

```
for(j=0;j<2;j++)
```

```
{k=0;
```

```
int temp=0;
```

```
int index=0;
```

```
while(((i-k)>=0)&&(i>0))
```

```
{
```

```
if(a[i][j].equals(s[i-k]))
```

```
{temp=1;
```

```
index=i-k;
```

```
}
```

```
k++;
```

```
}
```

```
if(temp==1)
```

```
{mt[index]+=1;
```

```
gf[index]+=Integer.parseInt(a[i][(j+2)]);
```

```
ga[index]+=Integer.parseInt(a[i][(3-j)]);
```

```
if((Integer.parseInt(a[i][j+2]))>(Integer.parseInt(a[i][3-j])))
```

```
pt[index]+=2;

else if(Integer.parseInt(a[i][j+2])==Integer.parseInt(a[i][3-
j]))

pt[index]+=1;

else

{pt[index]+=0;}

}

else

{

s[x]=a[i][j];

mt[x]+=1;

gf[x]=Integer.parseInt(a[i][(j+2)]);

ga[x]=Integer.parseInt(a[i][(3-j)]);

if((Integer.parseInt(a[i][j+2]))>(Integer.parseInt(a[i][3-j])))

pt[x]=2;

else if(Integer.parseInt(a[i][j+2])==Integer.parseInt(a[i][3-
j]))

pt[x]=1;

else

{pt[x]=0;}

x++;

}
```



```
}
```

```
}
```

```
int v=m;
```

```
int p[]=new int[n];
```

```
for(i=0;i<m;i++)
```

```
{ gd[i]=gf[i]-ga[i];
```

```
p[i]=pt[i];
```

```
}
```

```
int g[]=new int[n];
```

```
int b;
```

```
int r[]=new int[n];
```

```
for(i=0;i<m;++i)
```

```
{
```

```
for(j=0;j<(m-i-1);++j)
```

```
if(p[j]<p[j+1])
```

```
{
```

```
b=p[j];
```

```
p[j]=p[j+1];
```

```
p[j+1]=b;
```

```
}
```

```
}
```

```
int mp[]=new int[n];
```

```
int goalf[]=new int[n];
```

```
int goala[]=new int[n];
```

```
String tn[]=new String[n+n];
```

```
for(i=0;i<m;i++)
```

```
{
```

```
for(j=0;j<m;j++)
```

```
{ if(pt[i]==p[j])
```

```
{tn[j]=s[i];
```

```
mp[j]=mt[i];
```

```
goalf[j]=gf[i];
```

```
goala[j]=ga[i];
```

```
g[j]=gd[i];
```

```
}
```

```
}
```

```
}
```

```
for(i=m;i<n;i++)
{
    mp[i]=0;
    goalf[i]=0;
    goala[i]=0;
    g[i]=0;
    p[i]=0;
}
```

```
for(i=0;i<n;i++)
{for(j=0;j<m;j++)
{ if((z[i].equals(tn[j])))
{z[i]="0";
}
```

```
}
```

```
}
```

```
for(i=0;i<n;i++)
```

```
{  
    if(z[i]!="0")  
    {tn[v]=z[i];  
    v++;  
    }  
}  
  
for(i=0;i<n;i++)  
{  
    if(p[i]==p[i+1])  
    { if(g[i]>g[i+1])  
    {r[i]=i+1;  
    r[i+1]=i+2;  
    i=i+1;  
    }  
    else if(g[i]<g[i+1])  
    {r[i+1]=i+1;  
    r[i]=i+2;  
    i=i+1;  
    }  
    else  
    {
```

```
try
{
    int d=tn[i].compareToIgnoreCase(tn[i+1]);
    if(d>0)
    { s[i]=tn[i];
      tn[i]=tn[i+1];
      tn[i+1]=s[i];
    }
}

catch(NullPointerException npe)
{ npe.printStackTrace();
}

r[i]=i+1;
r[i+1]=i+1;
i=i+1;
}
}

else
r[i]=i+1;
}
```

```
System.out.print("\n\n\n\n\n");
```

```
System.out.print("\t\t\t" + "Football Points Table" + "\t\t");
```

```
System.out.print("\n\n");
```

```
System.out.println("Name" + "\t\t" + "Match" + "\t" + "GoalF" + "\t" + "Goal  
A" + "\t" + "GD" + "\t" + "Points" + "\t" + "Rank");
```

```
for(i=0;i<n;i++)
```

```
{
```

```
System.out.println(tn[i] + "\t\t" + mp[i] + "\t" + goalf[i] + "\t" + goala[i] + "\t"  
+ g[i] + "\t" + p[i] + "\t" + r[i]);
```

```
}
```

```
}
```

```
}
```

Python:

```
def func(matchdetails):
```

```
    goalfor=dict()
```

```
    goalag=dict()
```

```
    goaldif=dict()
```

```
    point=dict()
```

```

for i in range(len(matchdetails)):
    if matchdetails[i][0] in goalfor:
        goalfor[matchdetails[i][0]]+=matchdetails[i][2]
        goalag[matchdetails[i][0]]+=matchdetails[i][3]
    else:
        goalfor[matchdetails[i][0]]=matchdetails[i][2]
        goalag[matchdetails[i][0]]=matchdetails[i][3]
    if matchdetails[i][1] in goalfor:
        goalfor[matchdetails[i][1]]+=matchdetails[i][3]
        goalag[matchdetails[i][1]]+=matchdetails[i][2]
    else:
        goalfor[matchdetails[i][1]]=matchdetails[i][3]
        goalag[matchdetails[i][1]]=matchdetails[i][2]
    if(matchdetails[i][2] ==matchdetails[i][3]):
        if matchdetails[i][0] in point:
            point[matchdetails[i][0]]+=1
        else:
            point[matchdetails[i][0]]=1
        if matchdetails[i][1] in point:
            point[matchdetails[i][1]]+=1
        else:
            point[matchdetails[i][1]]=1

```

```

elif(matchdetails[i][2] > matchdetails[i][3]):
    if matchdetails[i][0] in point:
        point[matchdetails[i][0]] += 2
    else:
        point[matchdetails[i][0]] = 2
    elif(matchdetails[i][2] < matchdetails[i][3]):
        if matchdetails[i][1] in point:
            point[matchdetails[i][1]] += 2
        else:
            point[matchdetails[i][1]] = 2
    for i in goalfor:
        goaldif[i] = goalfor[i] - goalag[i]
    if i not in point:
        point[i] = 0
    res = {k: v for k, v in sorted(point.items(), key=lambda item:
item[1])}
    final = []
    values = res.values()
    if(len(set(values)) == len(values)):
        final.extend(list(res.keys()))
    else:
        count = 0

```



```
temp=0
for i in res:
    count+=1
    if(count==1):
        temp=i
    if(count !=1 and count<=len(res)):
        if(res[temp]==res[i]):
            if(goaldif[i]>goaldif[temp]):
                final.append(i)
            elif(goaldif[i]<goaldif[temp]):
                final.append(temp)
            temp=i
        else:
            if(goalfor[i]>goalfor[temp]):
                final.append(i)
            elif(goalfor[i]<goalfor[temp]):
                final.append(temp)
            temp=i
        else:
            a=sorted([i,temp])
            final.extend(a)
            temp=i
```

```

else:

if(res[temp]>res[i]):

final.append(temp)

temp=i

elif(res[temp]<res[i]):

final.append(i)

final.append(temp)

final=final[:-1]

return final

n=int(input("Enter the team size : "))

l=list(input("Enter the team : ").split(" "))

match=int(input())

matchdetails=[]

same=False

for i in range(match):

    matchlist=list(input("Enter the match details : ").split(" "))

    matchlist[2]=int(matchlist[2])

    matchlist[3]=int(matchlist[3])

    matchdetails.append(matchlist)

    if(matchlist[0]==matchlist[1]):

        same=True

if(same):

```

```
print("Invalid Input")

else:

    final=func(matchdetails)

    final=final[::-1]

    for i in final:

        if i in l:

            l.remove(i)

    sorted(l)

    final.extend(l[::-1])

    for i in final:

        print(i)
```

4. PROBLEM STATEMENT

Ron Wesley has been bit by a three headed snake and Harry Potter is searching for a potion. The Witch promises to tell the ingredients of the medicine if Harry can find equi pair of an array. Listen to the conversation between Harry The witch to know more about equi pairs.

Conversation:

The Witch: To find the equi pair, you must know how to find the slices first.

Harry: What is a slice?

The Witch: If Z is an array with N elements, a slice of indices (X, Y) is

$Z[X] + Z[X+1] \dots Z[Y]$

Harry : How can I use it to find equi pair?

The Witch: (a, b) is an equi pair if slice of $(0, a-1) = \text{slice of } (a+1, b-1)$

$= \text{slice of } (b+1, N-1)$ and $b > a+1$ and size of array > 4

Input Format:

An array of N integers delimited by white space

Output Format:

Print equi pair in first line in the format $\{a, b\}$

Print slices in the format $\{0, a-1\}, \{a+1, b-1\}, \{b+1, N-1\}$

OR

Print "Array does not contain any equi pair" if there are no equi pairs in the array

Constraints:

$Z_i \geq 0$ and $1 \leq i \leq N$

size of array $(N) > 4$

$b > (a+1)$

Example 1

Input:

8 3 5 2 10 6 7 9 5 2

Output:

Indices which form equi pair $\{3, 6\}$

Slices are { 0,2 }, { 4,5 }, { 7,9 }

Example 2

Input:

6 2 6 2 3 3 1 9

Output:

Array does not contain any equi pair.

C:

```
#include<stdio.h>
```

```
#define For(i,n) for(i=0;i<n;i++)
```

```
#define FOR(i,n) for(i=n-1;i>=0;i--)
```

```
#define nl printf("\n")
```

```
int main()
```

```
{
```

```
int a[100005],ind=0,i,n;
```

```
scanf("%d",&n);
```

```
for(i=0;i<n;i++){
```

```
    scanf("%d",&a[i]);
```

```
}
```

```
int k,j,fl=0;
```

```
For(i,n){
```

```
    if(i==0){}
```

```
    else a[i]+=a[i-1];
```

```

}

for(i=0;i<n;i++){
    for(j=i+2;j<n-2;j++){
        if(a[i]==a[j]-a[i+1] && a[j]-a[i+1]==a[n-1]-a[j+1]){
            printf("Indices which form equi pair { %d,%d }\n",i+1,j+1);
            printf("Slices are { 0,%d },{ %d,%d },{ %d,%d }\n",i,i+2,j,j+2,n-1);
            fl=1;
        }
    }
}

if(fl==0){
    printf("Array does not contain any equi pair\n");
}

return 0;
}

```

Java:

```

import java.util.*;

public class Main{

    public static void main(String[] args){

        Scanner ip=new Scanner(System.in);

        int[] a=new int[100];

        int i,n;
    }
}

```

```
n=ip.nextInt();  
for(i=0;i<n;i++){  
    a[i]=ip.nextInt();  
}  
  
int k,j,fl=0;  
  
for(i=0;i<n;i++){  
    if(i==0){}  
  
    else a[i]+=a[i-1];  
}  
  
for(i=0;i<n;i++){  
    for(j=i+2;j<n-2;j++){  
  
        if(a[i]==a[j]-a[i+1] && a[j]-a[i+1]==a[n-1]-a[j+1]){  
  
            System.out.printf("Indices which form equi pair { %d,%d  
}\n",i+1,j+1);  
  
            System.out.printf("Slices are { 0,%d },{ %d,%d },{ %d,%d  
}\n",i,i+2,j,j+2,n-1);  
  
            fl=1;  
  
        }  
  
    }  
  
}  
  
if(fl==0){  
  
    System.out.printf("Array does not contain any equi pair\n");  
}
```

```
}
}
}
```

Python:

```
n = int(input())

a = list(map(int,input().split()))

def check_eqipair(i,j):

    if sum(a[:i]) == sum(a[i+1:j]) == sum(a[j+1:n]):

        return 1

    return 0

for i in range(1,n-1):

    for j in range(i+1,n):

        x = check_eqipair(i,j)

        if x == 1:

            break

        if x == 1:

            break

        if x == 0:

            print('Array does not contain any equi pair')

    else:

        print("Indices which form equi pair {%d,%d}"%(i,j))

        print("Slices are {%d,%d},{%d,%d},{%d,%d}"%(0,i-1,i+1,j-1,j+1,n-1))
```


5. PROBLEM STATEMENT

Tahir and Mamta are working in a project in TCS. Tahir being a problem solver came up with an interesting problem for his friend Mamta.

Problem consists of a string of length N and contains only small case alphabets.

It will be followed by Q queries, in which each query will contain an integer P ($1 \leq P \leq N$) denoting a position within the string.

Mamta's task is to find the alphabet present at that location and determine the number of occurrences of same alphabet preceding the given location P .

Mamta is busy with her office work. Therefore, she asked you to help her.

Input Format:

- First line contains an integer N , denoting the length of string.
- Second line contains string S itself consists of small case alphabets only ('a' - 'z').
- Third line contains an integer Q denoting number of queries that will be asked.
- Next Q lines contains an integer P ($1 \leq P \leq N$) for which you need to find the number occurrence of character present at the P th location preceding P .

Output Format:

- For each query, print an integer denoting the answer on single line.

Constraints:

- $1 \leq N \leq 500000$
- S consisting of small case alphabets
- $1 \leq Q \leq 10000$

- $1 \leq P \leq N$

Example 1**Input:**

9

abacsddaa

2

9

3

Output:

3

1

C:

```
#include <stdio.h>
```

```
#include <string.h>
```

```
int main()
```

```
{
```

```
    int n;
```

```
    scanf("%d", &n);
```

```
    char str[n];
```

```
    scanf("%s", &str);
```

```
    int q;
```

```
    scanf("%d", &q);
```

```
    int qlines[q];
```

```
for(int i = 0; i<q; i++)  
{  
scanf("%d",&qlines[i]);  
}  
  
int counter[q];  
  
int result;  
  
for(int j = 0; j<q; j++) //how many times we have to iterate  
{  
  
int count1=0;  
  
int p=qlines[j]-1; //  
  
char r=str[p]; //char at pth position copied at char r  
  
for(int k = 0; k<qlines[j]-1; k++)  
{  
  
if(str[k] == r) //for comparing char  
{  
  
count1++;  
  
}  
  
}  
  
counter[j]=count1; //no of occurence stored at jth position  
  
}  
  
for(int i = 0; i<q; i++)  
  
printf("%d\n", counter[i]);
```

```
return 0;  
}
```

Java:

```
import java.util.*;  
  
public class Main{  
  
    public static void main(String[] args){  
  
        Scanner ip=new Scanner(System.in);  
  
        int n;  
  
        n=ip.nextInt();  
  
        char[] str=ip.next().toCharArray();  
  
        int q;  
  
        q=ip.nextInt();  
  
        int[] qlines=new int[q];  
  
        for(int i = 0; i<q; i++)  
  
        {  
  
            qlines[i]=ip.nextInt();  
  
        }  
  
        int[] counter=new int[q];  
  
        int result;  
  
        for(int j = 0; j<q; j++)  
  
        {  
  
            int count1=0;
```

```
int p=qlines[j]-1;

char r=str[p];

for(int k = 0; k<qlines[j]-1; k++)

{

    if(str[k] == r)

    {

        count1++;

    }

}

counter[j]=count1;

}

for(int i = 0; i<q; i++)

    System.out.printf("%d\n", counter[i]);

}

}
```

Python:

```
n=int(input()) #length of string

s=input() #string

q=int(input()) #test cases

p=[]

for i in range(0,q):

    p.append(int(input()))
```

```
for i in range(0,q):
```

```
    a=p[i]
```

```
    a=a-1
```

```
    c=0
```

```
    for x in range(0,a):
```

```
        if(s[x]==s[a]):
```

```
            c=c+1
```

```
    print(c)
```

1. CODU loves to play with string of brackets.

He considers string as a good string if it is balanced with stars. A string is considered as balanced with stars if string contains balanced brackets and between every pair of bracket i.e. between opening and closing brackets, there are at least 2 stars(*) present.

CODU knows how to check whether a string is balanced or not but this time he needs to keep a track of stars too. He decided to write a program to check whether a string is good or not. But CODU is not as good in programming as you are, so he decided to take help from you. Will you help him for this task?

You need to print Yes and number of balanced pair if string satisfies following conditions(string is good if it satisfies following 2 conditions):

1. The string is balanced with respect to all brackets.
2. Between every pair of bracket there is at least two stars.

However if string doesn't satisfies above conditions then print No and number of balanced pair in string as an output.

Constraints

4 <= String length <= 1000

Input Format

The first and only line of input contains a string of characters(a-z,A-Z), numbers(0-9), brackets('{', '[', '(', ')', ']', '}') and stars(*) .

Output

Print space separated "Yes" (without quotes) and number of balanced pair if string is good. Else print "No" (without quotes) and number of balanced pair.

Test Case**Explanation****Example 1****Input**

```
{**}
```

Output

Yes 1

Explanation

Here string contains one balanced pair {} and between this pair of bracket there are 2 stars present so the output is Yes with the count of balanced pair as 1.

Example 2**Input**

```
{**(**{**[**]})}
```

Output

Yes 4

Explanation

String has balanced brackets and also satisfies 2nd condition. So the output is Yes with count of balanced pair which is 4.

Example 3**Input**

```
**}xasd[**]sda231
```

Output

No 1

Explanation

In this case string is not balanced. So the output is No with the count of balanced pair as 1.

Solution:

```
import java.io.IOException;

import java.util.HashMap;

import java.util.Scanner;

import java.util.Stack;

import java.util.*;

class Main

{

    static HashMap<Character, Character> bMap;

    static

    {

        bMap=new HashMap<>();

        bMap.put('(',')');

        bMap.put('[',']');

        bMap.put('{','}');

    }

    static String isBalanced(String str)
```

```

{

    int count=0;

    String no="";

    char chs[]=new char[str.length()];

    for(int i=0;i<str.length();i++)

        chs[i]=str.charAt(i);


    Stack<Character> stack=new Stack<>();

    for(int i=0;i<str.length();i++)
    {

        char ch=chs[i];

        if(bMap.containsKey(ch))
        {

            stack.push(ch);

            count+=1;

        }

        else if(stack.isEmpty() ||
ch!=bMap.get(stack.pop()))
        {

            no="No";

        }

    }
}

```

```

        if(no=="No")
            return "No "+count;

        String ret="Yes "+count;

        return stack.isEmpty()?ret:"No "+count;
    }

    public static String RemoveUnwanted(String s)
    {

        String ret="";

        for(int i=0;i<s.length();i++)
        {

            char c=s.charAt(i);

            if( c=='(' || c==')' || c=='[' || c==']' ||
c=='{' || c=='}')

                {

                    ret+=c;

                }

        }

        return ret;

    }

    public static void main(String args[]) throws IOException
    {

        Scanner scanner=new Scanner(System.in);

```

```

scanner.skip("(\\r\\n|[\\n\\r\\u2028\\u2029\\u0085])?");

    String s=scanner.nextLine();

    if(s.length()<4 || s.length()>1000)

        System.exit(0);

    String ss=RemoveUnwanted(s);

    String result=isBalanced(ss);

    System.out.println(result);

}

}

```

2. Given two numbers n_1 and n_2 . Find prime numbers between n_1 and n_2 , then make all possible unique combinations of numbers from the prime numbers list you found in step 1.

From this new list, again find all prime numbers. Find smallest (a) and largest (b) number from the 2nd generated list, also count of this list. Consider smallest and largest number as the 1st and 2nd number to generate Fibonacci series respectively till the count (number of primes in the 2nd list).

Print the last number of a Fibonacci series as an output

Constraints

$2 \leq n_1, n_2 \leq 100$ $n_2 - n_1 \geq 35$

Input

One line containing two space separated integers n_1 and n_2 .

Output

Last number of a generated Fibonacci series.

Sample Input and Output

Example 1

Input

2 40

Output

13158006689

Explanation

1st prime list = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37]

Combination of all the primes = [23, 25, 27, 211, 213, 217, 219, 223, 229, 231, 32, 35, 37, 311, 313, 319, 323, 329, 331, 337, 52, 53, 57, 511, 513, 517, 519, 523, 529, 531, 537, 72, 73, 75, 711, 713, 717, 719, 723, 729, 731, 737, 112, 113, 115, 117, 1113, 1117, 1119, 1123, 1129, 1131, 1137, 132, 133, 135, 137, 1311, 1317, 1319, 1323, 1329, 1331, 1337, 172, 173, 175, 177, 1711, 1713, 1719, 1723, 1729, 1731, 1737, 192, 193, 195, 197, 1911, 1913, 1917, 1923, 1929, 1931, 1937, 232, 233, 235, 237, 2311, 2313, 2317, 2319, 2329, 2331, 2337, 292, 293, 295, 297, 2911, 2913, 2917, 2919, 2923, 2931, 2937, 312, 315, 317, 3111, 3113, 3117, 3119, 3123, 3129, 3137, 372, 373, 375, 377, 3711, 3713, 3717, 3719, 3723, 3729, 3731]

2nd prime list = [193, 3137, 197, 2311, 3719, 73, 137, 331, 523, 1931, 719, 337, 211, 23, 1117, 223, 1123, 229, 37, 293, 2917, 1319, 1129, 233, 173, 3119, 113, 53, 373, 311, 313, 1913, 1723, 317]

smallest (a) = 23

largest (b) = 3719

Therefore, the last number of a Fibonacci series i.e. 34th Fibonacci number in the series that has 23 and 3719 as the first 2 numbers is 13158006689

Example 2

Input

30 70

Output

2027041

Explanation

1st prime list=[31, 37, 41, 43, 47, 53, 59, 61, 67]

2nd prime list generated form combination of 1st prime list = [3137, 5953, 5347, 6761, 3761, 4337, 6737, 6131, 3767, 4759, 4153, 3167, 4159, 6143]

smallest prime in 2nd list=3137

largest prime in 2nd list=6761

Therefore, the last number of a Fibonacci series i.e. 14th Fibonacci number in the series that has 3137 and 6761 as the first 2 numbers is 2027041

Solution:

```
n1,n2=map(int,input().split())
```

```
list1=[]
```

```
for element in range(n1,n2+1):
```

```
    tpm=0
```

```
    if element==2:
```

```
        list1.append(element)
```

```
    else:
```

```
for i in range(2,element):
    if element%i==0:
        tpm=1
    if tpm==0:
        list1.append(element)
print(list1)

list2=[]

for element1 in list1:
    for element2 in list1:
        element1=str(element1)
        element2=str(element2)
        if element1!=element2:
            list2.append(element1+element2)

for i in range(len(list2)):
    list2[i]=int(list2[i])

print(list2)

list3=[]

for element in list2:
    tpm2=0

for i in range(2,element):
    if element%i==0:
        tpm2+=1
```

```
if tpm2==0:
    list3.append(element)
print(list3)
list4=[]
smallest=min(list3)
largest=max(list3)
count=len(list3)
a=smallest
b=largest
list4.append(a)
list4.append(b)
for i in range(count-2):
    c=a+b
    list4.append(c)
a=b
b=c
print(list4[len(list4)-1])
```

3. Given N three-digit numbers, your task is to find bit score of all N numbers and then print the number of pairs possible based on these calculated bit score.

Rule for calculating bit score from three digit number:

From the 3-digit number,

- extract largest digit and multiply by 11 then

- extract smallest digit multiply by 7 then
- add both the result for getting bit pairs.

Note: – Bit score should be of 2-digits, if above results in a 3-digit bit score, simply ignore most significant digit.

Consider following examples:

Say, number is 286

Largest digit is 8 and smallest digit is 2

So, $811 + 27 = 102$ so ignore most significant bit, So bit score = 02.

Say, Number is 123

Largest digit is 3 and smallest digit is 1

So, $311 + 71 = 40$, so bit score is 40.

Rules for making pairs from above calculated bit scores

Condition for making pairs are

- Both bit scores should be in either odd position or even position to be eligible to form a pair.
- Pairs can be only made if most significant digit are same and at most two pair can be made for a given significant digit.

Constraints

$N \leq 500$

Input Format

First line contains an integer N, denoting the count of numbers.

Second line contains N 3-digit integers delimited by space

Output

One integer value denoting the number of bit pairs.

Explanation

Example 1

Input

8 234 567 321 345 123 110 767 111

Output

3

Explanation

After getting the most and least significant digits of the numbers and applying the formula given in Rule 1 we get the bit scores of the numbers as:

58 12 40 76 40 11 19 18

No. of pair possible are 3:

40 appears twice at odd-indices 3 and 5 respectively. Hence, this is one pair.

12, 11, 18 are at even-indices. Hence, two pairs are possible from these three-bit scores.

Hence total pairs possible is 3.

Solution:

```
#include<stdio.h>
int bit_score(int n)
{
    int a, b, c, largest, smallest;
    int score;
    a = n%10; n/=10;
    b = n%10; n/=10;
    c = n%10; n/=10;
    largest = (a>b)?a:b;
    largest = (c>largest)?c:largest;
    smallest = (a<b)?a:b;
    smallest = (c<smallest)?c:smallest;
    score = largest*11 + smallest*7;
    score = score % 100;
    return score;
}
int findPairs (int score_array[], int N)
{
    int sig_dig[10], i, pairs = 0, msb;
    for(i=0; i<10; i++)
    {
        sig_dig[i] = 0;
    }
    for(i=0; i<N; i=i+2)
    {
        msb = score_array[i] / 10;
        for(int j =i+2; j<N; j=j+2)
        {
            if(msb == score_array[j]/10)
```

```

    {
        if(sig_dig[msb] < 2)
        {
            sig_dig[msb]++;
        }
    }
}
}
for(i=1; i<N; i=i+2)
{
    msb = score_array[i] / 10;
    for(int j =i+2; j<N; j=j+2)
    {
        if(msb == score_array[j]/10)
        {
            if(sig_dig[msb] < 2)
            {
                sig_dig[msb]++;
            }
        }
    }
    for(i=0; i<10; i++)
    {
        pairs = pairs + sig_dig[i];
    }
    return pairs;
}

int main()
{
    int N, i;
    int ip_array[501];
    int score_array[501];
    int pairs;
    scanf("%d",&N);
    for(i=0; i<N; i++)

```

```

{
    scanf("%d",&ip_array[i]);
}
for(i=0; i<N; i++)
{
    score_array[i] = bit_score(ip_array[i]);
}
pairs = findPairs(score_array, N);
printf("%d",pairs);
return 0;
}

```

4. A big group of students, starting a long journey on different set of vehicles need to fill petrol in their vehicles.

As group leader you are required to minimize the time they spend at the petrol pump to start the journey at the earliest. You will be given the quantity of petrol (in litres) that can be filled in each vehicle.

There are two petrol vending machines at the petrol pump. You need to arrange the vehicles in such a way that they take shortest possible time to fill all the vehicles and provide the time taken in seconds as output. Machine vends petrol @ 1litre/second.

Assume that there is no time lost between switching vehicles to start filling petrol.

Constraints

1<= Number of vehicles < 50.

0 <= Quantity of petrol required in any vehicle <= 200

Input Format

First line will provide the quantity of petrol (separated by space) that can be filled in each vehicle.

Output

Shortest possible time to fill petrol in all the vehicles.

Explanation

Example 1

Input

1 2 3 4 5 10 11 3 6 16

Output

31

Explanation

First Petrol vending machine will cater to vehicles taking – 16, 6, 4, 3, 2 litres of petrol (Total 31 sec)

Second machine will cater to vehicles taking – 11, 10, 5, 3, 1 litres of petrol (Total 30 sec)

Example 2**Input**

25 30 35 20 90 110 45 70 80 12 30 35 85

Output

335

Explanation

First Petrol vending machine will cater to vehicles taking – 80, 45, 35, 30, 25, 12, 85, 20 litres of petrol.

Second machine will cater to vehicles taking – 90, 70, 35, 30, 110 litres of petrol. Since second machine will take more time, total time to fill petrol in all vehicles will be 335 seconds.

Solution:

```
#include <stdio.h>
int main(void) {
    int n=0, x, i, j, s = 0;
    int a[51];
    while((scanf("%d",&x))!=-1)
    {
        a[n++]=x;
        s += x;
    }
    int d[n+1][s+1];
    for(i=0 ; i<=n ; i++)
        d[i][0] = 1;
    for(i=1 ; i<=s ; i++)
        d[0][i] = 0;

    for(i=1 ; i<=n ; i++)
        for(j=1 ; j<=s ; j++)
```

```

    {
        d[i][j] = d[i-1][j];
        if(a[i-1] <= j)
            d[i][j] = d[i][j] | d[i-1][j - a[i-1]];
    }
    int an = s;
    for(i=s/2 ; i>=0 ; --i)
        if(d[n][i])
        {
            an = s - i;
            break;
        }
    printf("%d",an);
    return 0;
}

```

5. Problem Description

In a crossover fantasy universe, Houin Kyoma is up in a battle against a powerful monster Nomu that can kill him in a single blow. However being a brilliant scientist Kyoma found a way to pause time for exactly M seconds. Each second, Kyoma attacks Nomu with certain power, which will reduce his health points by that exact power. Initially Nomu has H Health Points. Nomu dies when his Health Points reach 0. Normally Kyoma performs Normal Attack with power A . Besides from Kyoma's brilliance, luck plays a major role in events of this universe. Kyoma's Luck L is defined as probability of performing a super attack. A super attack increases power of Normal Attack by C . Given this information calculate and print the probability that Kyoma kills Nomu and survives. If Kyoma dies print "RIP".

Constraints

$0 < T \leq 50$

$1 \leq A, H, C, L1, L2 \leq 1000$

$1 \leq M \leq 20$.

$L1 \leq L2$

Input Format

First line is integer T denoting number of test cases.

Each test case consist of single line with space separated numbers A H L1 L2 M C. Where luck L is defined as $L1/L2$. Other numbers are, as described above.

Output

Print probability that Kyoma kills Nomu in form $P1/P2$ where $P1 \leq P2$ and $\gcd(P1, P2) = 1$. If impossible, print "RIP" without quotes.

Explanation

Example 1

Input

2

10 33 7 10 3 2

10 999 7 10 3 2

Output

98/125

RIP

Solution:

```
from math import gcd, factorial
```

```
def ncr(n, r):
```

```
    return factorial(n) / (factorial(n - r) * factorial(r))
```

```
for i in range(int(input())):
```

```
    a, h, l1, l2, m, c = [int(i) for i in input().split()]
```

```
    num = 0
```

```

den = pow(l2,m)

if m*(a+c) < h:
    print("RIP")
else:
    k,z = 0,m*a
    while z < h:
        z += c
        k += 1
    for i in range(k,m+1):
        if i == 0:
            num += pow(l2 - l1,m)
        elif i == m:
            num += pow(l1,i)
        else:
            num += (pow(l1,i)*pow(l2-l1,m-i)*ncr(m,i))
    x = gcd(int(num),den)
    print(str(int(num/x))+"/"+str(int(den/x)))

```

6. You are a teacher in reputed school. During Celebration Day you were assigned a task to distribute Cadbury such that maximum children get the chocolate. You have a box full of Cadbury with different width and height. You can only distribute largest square shape Cadbury. So if you have a Cadbury of length 10 and width 5,

then you need to break Cadbury in 5X5 square and distribute to first child and then remaining 5X5 to next in queue

Constraints $0 < P < Q < 1501$ $0 < R < S < 1501$ **Input Format**

First line contains an integer P that denotes minimum length of Cadbury in the box

Second line contains an integer Q that denotes maximum length of Cadbury in the box

Third line contains an integer R that denotes minimum width of Cadbury in the box

Fourth line contains an integer S that denotes maximum width of Cadbury in the box

Output

Print total number of children who will get chocolate.

Explanation**Example 1**

Input

5

7

3

4

Output

24

Explanation

Length is in between 5 to 7 and width is in between 3 to 4.

So we have 5X3, 5X4, 6X3, 6X4, 7X3, 7X4 type of Cadbury in the box.

If we take 5X3 :

First, we can give 3X3 square Cadbury to 1st child .Then we are left with 3X2. Now largest square is 2X2 which will be given to next child.

Next, we are left with two 1X1 part of Cadbury which will be given to another two children.And so on

Solution:

```
#include<stdio.h>

int no_of_children(int row, int col)
{
    int count=0;
    int total = row * col;
    while(row && col)
    {
        count++;
        if(row>col)
            row = row - col;
        else
            col = col - row;
    }
    return count;
}

int main()
{
    int sum=0;
    int minlen,maxlen,minwid,maxwid;
    scanf("%d\n%d\n%d\n%d",&minlen,&maxlen,&minwid,&maxwid);
    if(0<minlen<1501 && 0<maxlen<1501 && 0<minwid<1501 &&
    0<maxwid<1501)
```

```

{
    for(int i=minlen;i<=maxlen;i++)
    {
        for(int j=minwid;j<=maxwid;j++)
        {
            sum = sum + no_of_children(i,j);
        }
    }
    printf("%d",sum);
}

return 0;
}

```

7. There are T tubs of water, numbered from 1 to T . Initially there is a very few litres of water in each tub T. Each water tub has Two taps attached to it. Incoming Tap with speed x lit /sec & Outgoing Tap with speed y lit /sec . Let water(i) denoting the final vol. of water in ith tub. Karan wants to attain such a situation that $\text{water}(i) < \text{water}(i+1)$ for $1 \leq i \leq T$. i.e. water in each tub must be less than water in the tub next to it. He wants to do this as quickly as possible. Your task is to find out and tell Karan , what is the minimum no. of seconds required to attain in this situation.

Input Format

First line will contain the number of tubs, denoted by T Next T lines contain a tuple with 3 integers delimited by white space. The tuple contents are

1. W_i - Vol. of water is present at beginning in ith tub (in litres)

2. x - Denoting speed of incoming tap of i th tub (in lit /sec)

3. y - Denoting speed of outgoing tap of i th tub (in lit /sec)

Output Format

Minimum time in seconds, needed to arrange water in T tubs, in ascending order of volume.

Constraints

- $2 \leq T \leq 100000$
- $0 \leq W \leq 1000000000$ (1 billion)
- for each tub $1 \leq x \leq 10000$
- for each tub $1 \leq y \leq 10000$

for each tub A tap can be used only for integral number of sec i.e. you cannot use a tap for float(0.3 or 0.5 sec) . It can be used either for 1,2,3,4.... sec Capacity of each tub is infinite.

Vol. of water in any tub cannot be less than 0 at any point of time.
Any no. of taps can be turn on / off simultaneously.

Sample Input and Output

Example 1

Here we have Three tubs with information each of them as follow

Tub Number	Initial Volume	Incoming Tap speed	Outgoing Tap speed
Tub 1	2	3	3
Tub 2	3	4	5
Tub 3	3	5	5

Initially tub 2 and 3 have same Vol. of water. So Karan will just turn on the Incoming Tap at Tub 3 for 1 sec. After 1 sec the water in Third tub will be 8 lit . Since $2 < 3 < 8$, Therefore answer is 1 sec .

Example 2

Here we have Three tubs with following each of them as follow

Tub Number	Initial Volume	Incoming Tap speed	Outgoing Tap speed
Tub 1	6	2	1
Tub 2	4	1	3
Tub 3	4	1	4

As we can see that it is not possible to do the task in one sec . But it can be done in 2 sec as shown below .

Turn on the outgoing tap at tub 1. So after 2 sec water level will reduce to 4 lit . Turn on the incoming tap at tub 2 only for 1 sec . So that water level will be 5 lit . Turn on the incoming tap at tub 3 for 2 sec . So that water level will be 6 lit . Since $4 < 5 < 6$, therefore answer is 2 sec .

Solution

```
#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

void main()

{

    int i,j=0,n,t=0;

    long *w,*x,*y;

    scanf("%d",&n);

    w=(long*)calloc(n,sizeof(long));

    x=(long*)calloc(n,sizeof(long));
```

```
y=(long*)calloc(n,sizeof(long));  
for(i=0;i<n;i++)  
{  
    scanf("%d%d%d",&w[i],&x[i],&y[i]);  
}  
for(i=0;i<n;i++)  
{  
    if(w[i]<w[i+1])  
    {  
        j=0;  
        continue;  
    }  
    else  
    {  
        goto loop;  
    }  
}  
if(j==0)  
{  
    goto ans;  
}  
loop: for(i=0;i<n;i++)
```

```
{  
    t=1;  
    cmp: if(w[i]-y[i]>0&& i==0)  
    {  
        w[i]=w[i]-y[i];  
    }  
    else if(w[i]-y[i]>0 && w[i]-y[i]>w[i-1])  
    {  
        w[i]=w[i]-y[i];  
    }  
    if(w[i+1]<=w[i])  
    {  
        w[i+1]=w[i+1]+x[i+1];  
        if(w[i+1]<=w[i])  
        {  
            t++;  
            goto cmp;  
        }  
    }  
    if(j<=t)  
    j=t;  
}
```

```
printf("%d",j);

getch();

}
```

8. On a busy road, multiple cars are passing by. A simulation is run to see what happens if brakes fail for all cars on the road. The only way for them to be safe is if they don't collide and pass by each other. The goal is to identify whether any of the given cars would collide or pass by each other safely around a Roundabout. Think of this as a reference point O (Origin with coordinates (0,0)), but instead of going around it, cars pass through it.

Considering that each car is moving in a straight line towards the origin with individual uniform speed. Cars will continue to travel in that same straight line even after crossing origin. Calculate the number of collisions

that will happen in such a scenario.

Note : – Calculate collisions only at origin. Ignore the other collisions. Assume that each car continues on its respective path even after the collision without change of direction or speed for an infinite distance.

Output Format

A single integer Q denoting the number of collisions at origin possible for given set of cars.

Constraints

$1 \leq T \leq 100$

$1 \leq N \leq 5000$

Sample Input and Output

Input

```
5
5 12 1
16 63 5
-10 24 2
7 24 2
-24 7 2
```

Output

```
4
```


Explanation

Let the 5 cars be A, B, C, D, and E respectively.

4 Collisions are as follows –

- 1) A & B.
- 2) A & C.
- 3) B & C.
- 4) D & E.

Solution

```
import math

C=int(input())

T={}

collision=0

for i in range(C):

    x,y,v=list(map(int,input().split()))

    t=math.sqrt(((x/v)**2+(y/v)**2))

    if(T.get(t)==None):

        T[t]=1

    else:

        T[t]=T[t]+1

for keys in T:

    if(T[keys]!=1):

        collision=collision+(T[keys]*(T[keys]-1))/2

print(int(collision))
```

9. Suppose You are a System Administrator (SA) . You have access to unlimited no. of servers with same amount of CPU (#of cores) & Memory (GB).

On these servers, you have to pack number of applications X , deployed as VMs such that the total cost of ownership without affecting the application performance, is the smallest . Application resource requirements are given in the following format
CPU Utilization of computer in percentage for a single core system
Memory Allocated in MB

You have to pack the applications on the server in such a way that its resource requirement is fully met on that server i.e. we cannot deploy the same application across different servers.

Each server has a constant initial cost in terms of

1. Reserved CPU in % for a single core system
2. Reserved Memory in Mega Byte MB

From the following equation we can cost of the server:

Cost per hr = constant initial cost + $(1.5/10000)(\text{cpu_utiliz_os_server}^3) + 0.5(\text{mem_utiliz_of_server})$

where $0 \leq \text{cpu_utiliz_os_server} \leq 100$ and constant initial cost = 100

$\text{mem_utiliz_of_server} \leq 100$ and constant initial cost = 100

Your task is to find the minimum cost per hr (hour) for running applications.

Constraints

- $1 \leq \text{number of cores} \leq 10$
- $1 \leq \text{memory in GB} \leq 10$
- $0 \leq \text{number of VMs} \leq 100$
- $1 \text{ GB} = 1024 \text{ MB}$

Input Format

First line

It provides the no. of cores, C, in each server that host application

Second line

It provides the amount of memory, M, in GB in each server

Third line

It provides no. of applications, X , to be packed on servers

Next, $X * 2$ lines consists of the following information, on one line each

CPU (Central Processing Unit) is requirement in percentage of single core for current application

Memory requirement in MB for current application

Last 2 lines

consists of the following information

Reserved CPU (Central Processing Unit) in percentage of a single core system

Reserved Memory in Mega Byte

Output Format

Least cost per hr for hosting all applications , rounded up to the next integer

Input

2

1

4

29.72

366

25.53

163

28.98

206

26.64

506

11.21

176

Output

289

Input

3

1

4

26.66

451

25.8

294

26.81

207

26.77

192

8.03

184

Output

277

Solution

```
#include<stdio.h>
```

```
main()
```

```
{
```

```
long long int Total1, Total2, Ans;
```

```
long int C[1000], M[1000], cpu_requirement[1000],  
memory_requirement[1000], cpu_utilization[5000],  
memory_utilization[5000], reserved_cpu[1000],  
reserved_memory[1000];
```

```
int K[1000], k, s, i;
```

```
printf("Enter the total number of servers \n");
```

```
scanf("%d", &i);
```

```
for(s=0; s<i; s++);
```

```
{
```

```
printf(" Enter the number of cores, C, in each server \n");
```

```
scanf("%ld",&C[s]);
```

```
printf("Enter the amount of memory, M, in gigabytes in each server  
\n");
```

```
scanf("%ld",&M[s]);
```

```
printf("Enter number of applications, K, to be packed on server \n");
```

```
scanf("%d",&K[s]);
```

```
up: for(k=0; k<K[s]; k++)
```

```
{
```

```
printf ("Enter the CPU requirement in percentage of single core for  
current application \n");
```

```
scanf("%ld",&cpu_requirement[k]);
```

```
printf("Enter Memory requirement in megabytes for current  
application \n");
```

```
scanf("%ld",&memory_requirement[k]);
```

```
cpu_utilization[s]=0;
```

```
cpu_utilization[s]=cpu_utilization[s]+ cpu_requirement[k];
```

```
cpu_utilization[s]=cpu_utilization[s]*C[s];
```

```
memory_utilization[s]=0;
```

```
memory_utilization[s]=memory_utilization[s]+memory_requirement  
[k];
```

```
}
```

```
if(memory_utilization[s]>M[s])
```

```
{  
  
printf("The memory limit is exceeded, please enter proper memory  
requirement \n");  
  
Goto up;  
  
}  
  
printf("Enter the Reserved CPU in percentage of a single core system  
\n");  
  
scanf("%ld",&reserved_cpu[s]);  
  
printf("Enter the Reserved Memory in MB \n");  
  
scanf("%ld",&reserved_memory[s]);  
  
}  
  
for(s=0; s<i; s++)  
{  
  
Total1=0;  
  
Total1=reserved_cpu[s]+cpu_utilization[s]+Total1;  
  
Total2=0;  
  
Total2=reserved_memory[s]+memory_utilization[s]+Total2;  
  
}
```

```

Ans = 100 + (( 1.5/10000 ) * pow(Total1,3)) + (0.5*(Total2));

printf("Least cost per hour for hosting all applications is %lld",Ans);

}

```

10. Dr. Vishnu is opening a new world class hospital in a small town designed to be the first preference of the patients in the city. Hospital has N rooms of two types - with TV and without TV, with daily rates of R1 and R2 respectively.

However, from his experience Dr. Vishnu knows that the number of patients is not constant throughout the year, instead it follows a pattern.

The number of patients on any given day of the year is given by the following formula :

$$(6-M)^2 + |D-15|$$

where

M is the number of month (1 for jan, 2 for feb ...12 for dec) and

D is the date (1,2...31).

All patients prefer without TV rooms as they are cheaper, but will opt for with TV rooms only if without TV rooms are not available.

Hospital has a revenue target for the first year of operation. Given this target and the values of N, R1 and R2 you need to identify the number of TVs the hospital should buy so that it meets the revenue target. Assume the Hospital opens on 1st Jan and year is a non-leap year.

Input Format

First line provides an integer N that denotes the number of rooms in the hospital

Second line provides two space-delimited integers that denote the rates of rooms with TV (R1) and without TV (R2) respectively .

Third line provides the revenue target

Output Format

Minimum number of TVs the hospital needs to buy to meet its revenue

target. If it cannot achieve its target, print the total number of rooms in the

hospital.

Constraints

Hospital opens on 1st Jan in an ordinary year

1. $5 \leq \text{Number of rooms} \leq 100$
2. $500 \leq \text{Room Rates} \leq 5000$
3. $0 \leq \text{Target revenue} < 90000000$

Sample Input and Output

Example 1

Input

20
1500 1000
7000000

Output

14

Explanation

Using the formula, number of patients on 1st Jan will be 39, on 2nd Jan will be 38 and so on. Considering there are only twenty rooms and rates of both type of rooms are 1500 and 1000 respectively, we will need 14 TV sets to get revenue of 7119500. With 13 TV sets Total revenue will be less than 7000000 .

Example 2

Input

10

1000 1500

10000000

Output

10

Explanation

In the above example, the target will not be achieved, even by equipping all the rooms with TV. Hence, the answer is 10 i.e. total number of rooms in the hospital.

Solution:

```
N=int(input())
```

```
r1,r2=map(int,input().split())
```

```
Target=int(input())
```

```
l1,l2=[],[]
```

```
cost,final=0,0
```

```
l=[31,28,31,30,31,30,31,31,30,31,30,31]
```

```
for j in range(len(l)):
```

```
    for k in range(1,l[j]+1):
```

```
        l1.append((6-(j+1))**2+abs(k-15))
```

```
    l2.append(l1)
```

```
    l1=[]
```

```
for i in range(N+1):
```

```
for j in l2:
    for k in j:
        if(k>=N):
            t=N-i
            cost=cost+(i*r1+t*r2)
        else:
            h=N-i
            t=k-h
            if(t<=0):
                cost=cost+(k*r2)
            else:
                cost=cost+(t*r1+h*r2)
    final=final+cost
    cost=0
if(final>=Target):
    print(i)
    break
else:
    final=0
else:
    print(N)
```

11. The company named "One Egg" which supplies eggs to retailers. They have M classes of eggs. Each class can have N no. of eggs (N can be same or it can vary class to class). They accept an order via mail for X eggs. In response, they confirm if they can supply the eggs with a "Thank you" note as well as the no. of eggs or with a "Sorry" note and the no. of eggs they can supply. They also mention the breakdown of eggs by class they will supply. The ordered eggs are adjusted against the different classes with the most no. of eggs adjusted first then the balance is adjusted against the second highest and so on.

The company named "One Egg" is a bit superstitious as well . If the no. of eggs ordered is greater than or equal to the total no. of eggs in stock then they retain 1 egg and responds back with the "Sorry"

Note : If the classes have same number of eggs then class entered first should be selected to adjust. with total number of eggs in stock minus 1 and breakdown of eggs by class.

Input Format

First line

It contains 2 space-separated integers describing the respective values of M & X, the no. of eggs ordered .

The following M lines contain an integer each indicating the no. of eggs available in each class

Where M : the number of classes of eggs

Output Format

First line

Described if X is less than total no. of Eggs then Print " Thank you, your order for X eggs is accepted" Else if X is greater than or equal to total no. of Eggs then print " Sorry, we can only supply (total number of Eggs in stock -1) eggs" Then M lines with three columns: 1st column - no of eggs available in each class 2nd column - Eggs

allocated against each class for that order 3rd column - Balance Eggs
against each class

Constraints

- $1 \leq M \leq 20$
- $N \geq 1$
- $X \geq 1$

Sample Input and Output

Example 1

Input

5 150
50
15
80
10
5

Example 2

Input

4 250
80
50
70
20

Output

Sorry, we can only supply 219 eggs

8 0	80	0
5 0	50	0

```
7 0    70    0
2 0    19    1
```

Explanation

Total order of 250 eggs was greater than the total no of eggs $80+50+70+20 = 220$. Therefore the sorry message. 250 was first adjusted against Class with first highest no of eggs 80. Therefore Balance of $250-80 = 170$ was adjusted against second highest class of 70. Therefore Balance of $170-70 = 100$ was then adjusted against 50. Therefore Balance of $100-50 = 50$ then adjusted against 20. Where Balance is greater than last class of egg all but one egg is left in that last class.

```
#include <stdio.h>

int main() {

int m,x,i,a[1000],sum=0,s;

scanf("%d %d",&m,&x);

for(i=0;i<m;i++)

{

scanf("%d",&a[i]);

sum=sum+a[i];

}

if(sum>x)

    printf("Thank you, your order for %d eggs are accepted\n",x);

else

{
```

```
printf("Sorry, we can only supply %d eggs\n",sum-1);

x=sum-1;

}

for(i=0;i<m;i++)

{

    if(x>=a[i])

    {

        printf("%d\t%d\t%d\n",a[i],a[i],0);

        x=x-a[i];

    }

    else if(x<a[i])

    {

        s=a[i]-x;

        printf("%d\t%d\t%d\n",a[i],x,s);

        x=0;

    }

    else if(x==0)

        printf("%d\t%d\t%d\n",a[i],0,a[i]);

}

return 0;
```

}

12. Given a sequence of distinct numbers $a_1, a_2, a_3, \dots, a_n$, an inversion occurs if there are indices $i < j$ such that $a_i > a_j$

For example in the sequence 2 1 4 3 there are 2 inversions (2 1) and (4 3)

The input will be main sequence of N positive integers. From this sequence, a derived sequence

Will be obtained using the following rule. The output is the number of inversions in the derived sequence

Rules for forming derived sequence

The derived sequence is formed by counting the number of 1's bits in the binary representation of the corresponding number in the input sequence

Thus, if the input is 3,4,8 the binary representation of the numbers are 11100 and 1000.

The derived sequence is the number of 1's in the binary representation of the number in the input sequence and is 2,1,1

Constraints:

$N \leq 50$

Integers in sequence $\leq 10^7$

Input Format

First line of input will have a single integer, which will give N .

The next line will consists of comma separated string of N integers, which is the main sequence

Output

The number of inversions in the derived sequence formed from the main sequence.

Explanation

Example 1

Input

5

55 53 88 27 33

Output

8

Explanation:

The number of integers is 5, as specified in the first line. The given sequence is 55, 53, 88, 27, 33.

The binary representation is 110111,110101, 1011000, 11011, 100001 and 100001. The derived sequence is 5,4,3,4,2, 4,3,4,2 (corresponding to the number of 1s bits). The number of inversions in this is 8, namely (5,4),(5,3),(5,4),(5,2),(4,3),(4,2),(3,2),(4,2). Hence the output is 8.

Example 2

Input

8

120 21 47 64 72 35 18 98

Output

15

Solution

```
def decimalToBinary(n):
    return bin(n).replace("0b", "")
N=int(input())
Numbers=[]
A=[]
c=0
if(N<=50):
    Numbers=list(map(int,input().split(",")))
    for _i_ in Numbers:
        BinaryNo=decimalToBinary(_i_)
        n=BinaryNo.count('1')
        A.append(n)
    for i in range(N):
        for j in range(i+1,N):
            if(A[i]>A[j]):
                c=c+1
    print(c)
```

13. A ceremony where a Bride chooses her Groom from an array of eligible bachelors is called Swayamvar. But this is a Swayamvar with

difference. An array of Bride-to-be will choose from an array of Groom-to-be.

The arrangement at this Swayamvar is as follows

- Brides-to-be are organized such that the most eligible bachelorette will get first chance to choose her Groom. Only then, the next most eligible bachelorette will get a chance to choose her Groom
- If the initial most eligible bachelorette does not get a Groom of her choice, none of the Brides-to-be have any chance at all to get married. So unless a senior bachelorette is out of the “queue”, the junior bachelorette does not get a chance to select her Groom-to-be
- Initial state of Grooms-to-be is such that most eligible bachelor is at the head of the “queue”. The next most eligible bachelor is next in the queue. So on and so forth.
- Now everything hinges on the choice of the bachelorette. The most eligible bachelorette will now meet the most eligible bachelor.
- If bachelorette selects the bachelor, both, the bachelorette and the bachelor are now Bride and Groom respectively and will no longer be a part of the Swayamvar activity. Now, the next most eligible bachelorette will get a chance to choose her Groom. Her first option is the next most eligible bachelor (relative to initial state)
- If the most eligible bachelorette passes the most eligible bachelor, the most eligible bachelor now moves to the end of the queue. The next most eligible bachelor is now considered by the most eligible bachelorette. Selection or Passing over will have the same consequences as explained earlier.
- If most eligible bachelorette reaches the end of bachelor queue, then the Swayamvar is over. Nobody can get married.
- Given a mix of Selection or Passing over, different pairs will get formed.

The selection criteria is as follows

1. Each person either drinks rum or mojito. Bride will choose groom only if he drinks the same drink as her.

Note : There are equal number of brides and grooms for the swayamvar.

Tyrion as the hand of the king wants to know how many pairs will be left unmatched. Can you help Tyrion?

Input Format

First line contains one integer N , which denotes the number of brides and grooms taking part in the swayamvar.

Second line contains a string in lowercase, of length N containing initial state of brides-to-be.

Third line contains a string in lowercase, of length N containing initial state of grooms-to-be. Each string contains only lowercase 'r' and 'm' stating person at that index drinks "rum"(for 'r') or mojito(for 'm').

Output Format

Output a single integer denoting the number of pairs left unmatched.

Timeout

1

Constraints

- $0 < P < Q < 1501$
- $0 < R < S < 1501$

Sample Input and Output

Input

4

rrmm

mrmm

Output

0

Explanation

The bride at first place will only marry groom who drinks rum. So the groom at first place will join the end of the queue. Updated groom's queue is "rmm".

Now the bride at first place will marry the groom at first place. Updated bride's queue is "rmm" and groom's queue is "mrm".

The process continues and at last there are no pairs left. So answer is 0.

Solution:

```
number_of_brides_and_grooms = int(input())
```

```
brides = input()
```

```
grooms = input()
```

```
i = 0
```

```
while i < number_of_brides_and_grooms:
```

```
    index = grooms.find(brides[0])
```

```
    if index < 0:
```

```
        break
```

```
    grooms = grooms[index+1:]+grooms[:index]
```

```
    brides = brides[1:]
```

```
i += 1
```

```
print(len(bridges),end="")
```