

Quiz questions

0. Title: Range of char

Description: What is the range of character bucket acquire

0.1 Byte

1.4 Byte

2.8 Byte

3.16 Byte

1. Title: Order of precedence

Description: "What is the order of precedence for int, float, char and bool

"

0.char>float>int>bool

1.float>int>char>bool

2.bool>float>int>char

3.int>float>char>bool

2. Title: Predict Output 1

Description: "/* The output of the following program is ? */

- 1) `#include<iostream>`
- 2) `using namespace std;`
- 3) `int main(){`
- 4) `signed CodingBlocks = 9;`

```

5)    signed Nagarro = A;
6)    signed char HackerBlocks = 'A';
7)    cout<<CodingBlocks<<endl;
8)    cout<<HackerBlocks<<endl;
9)    return 0;
    }
"

```

0.A 9

1.Error in line 4

2.Error in line 5 //signed nagarro is not declare

3.9 A

3. Title: Predict Output 2

Description: "/* What will be the output of the following program ? */

```cpp

#include<iostream>

using namespace std;

int main(){

signed CodingBlocks = 9;

short double Nagarro = 8.8;

signed char HackerBlocks = 'A';

cout<<CodingBlocks<<endl;

cout<<Nagarro<<endl;

cout<<HackerBlocks<<endl;

```
 return 0;

}"

...

```

0.9 8.8 A

1."Run Time Error "

2.Error due to Nagarro

3.Error Due to Coding Blocks

4. Title: Predict Output 3

Description: ``cpp

```
"/* Output ? */
```

```
#include<iostream>
```

```
using namespace std;
```

```
int main(){
```

```
 long signed CodingBlocks = 2017;
```

```
 short unsigned BOSS1 = -2018;
```

```
 unsigned BOSS2 = -2019;
```

```
 int BOSS3 = -2020;
```

```
 long long unsigned BOSS4 = -2021;
```

```
 short unsigned Nagarro = 2018.9;
```

```
 long signed HackerBlocks = 'A';
```

```
 cout<<CodingBlocks<<endl;
```

```
 cout<<BOSS1<<endl<<BOSS2<<endl<<BOSS3<<endl<<BOSS4<<endl;
```

```
 cout<<Nagarro<<endl;
```

```
cout<<HackerBlocks<<endl;
```

```
return 0;
```

```
}"
```

```
...
```

0.2017 2^16 - 2018 2^32 -2019 -2020 2^64 - 2021 2018 65

1.2017 2018 2019 2020 2021 2018.9 A

2.2017 2^32 - 2018 2^32 -2019 -2020 2^32 - 2021 2018 65

3.2017 2^32 - 2018 2^32 -2019 -2020 2^64 - 2021 2018 64

### Problem Name: Simple Input

Problem Difficulty: 1

Problem Constraints: All numbers input are integers between -1000 and 1000.

Problem Description:

Given a list of numbers, stop processing input after the cumulative sum of all the input becomes negative.

Input Format: A list of integers to be processed

Sample Input: 1

2

88

-100

49

Output Format: Print all the numbers before the cumulative sum become negative.

Sample Output: 1

2

88

```

#include<iostream>
using namespace std;

int main() {

 int sum=0;
 while(true)
 {
 int n;
 cin>>n;
 sum=sum+n;
 if(sum>=0)
 {
 cout<<n<<endl;
 }
 else{
 break;
 }
 }
 return 0;
}

```

### Problem Name: Revising Quadratic Equations

Problem Difficulty: 1

Problem Constraints:  $-100 \leq a, b, c \leq 100$

Problem Description:

Given coefficients of a quadratic equation , you need to print the nature of the roots (Real and Distinct , Real and Equal or Imaginary) and the roots. <br>

If Real and Distinct , print the roots in increasing order. <br>

If Real and Equal , print the same repeating root twice <br>

If Imaginary , no need to print the roots.

Note : Print only the integer part of the roots.

Input Format: First line contains three integer coefficients a,b,c for the equation  $ax^2 + bx + c = 0$ .

Sample Input: 1 -11 28

Output Format: Output contains one/two lines. First line contains nature of the roots .The next line contains roots(in non-decreasing order) separated by a space if they exist. If roots are imaginary do not print the roots.

Output the integer values for the roots.

Sample Output: Real and Distinct

4 7

```
#include<iostream>
#include<math.h>
#include<cmath>
using namespace std;
int main()
{
 int a,b,c;

 cin>>a>>b>>c;
 float d;
 float d1;
 d=b*b-4*a*c;
 d1=sqrt(d);
 if(d==0)
 {
 cout<<"Real and Equal";

 }
 else if(d>=0)
 {

 cout<<"Real and Distinct";
 }
 else if(d<0)
 {
 cout << "Imaginary";

 }
 cout<<endl;
 if(d>=0)
 {
```

```

 int x1,x2;
 x1=(-b-d1)/2*a;
 x2=(-b+d1)/2*a;
 cout<<x1<<" "<<x2;
}

return 0;
}

```

Problem Name: **Print reverse**

Problem Difficulty: 1

Problem Constraints:  $0 \leq N \leq 1000000000$

Problem Description:

Take N as input, Calculate it's reverse also Print the reverse.

Input Format:

Sample Input: 123456789

Output Format:

Sample Output: 987654321

```

#include<iostream>
using namespace std;
int main()
{

 int n;
 cin>>n;
 int rev=0;
 while(n>0)
 {

 int rem=n%10;
 rev=rev*10+rem;
 n=n/10;
 }
}

```

```
cout<<rev;
}
```

Problem Name: Von Neuman Loves Binary

Problem Difficulty: 1

Problem Constraints:  $N \leq 1000$

Digits in binary representation is  $\leq 16$ .

Problem Description:

Given a binary number ,help Von Neuman to find out its decimal representation.

For eg 000111 in binary is 7 in decimal.

Input Format: The first line contains N , the number of binary numbers.

Next N lines contain N integers each representing binary representation of number.

Sample Input: 4

101

1111

00110

111111

Output Format: N lines,each containing a decimal equivalent of the binary number.

Sample Output:

5

15

6

63

```
#include<iostream>
using namespace std;
int main() {

 int N;
 int no;
```



```

 cin>>no;
 while(no>0)
 {
 cin>>N;

 int pow=1;
 int ans=0;
 while(N>0)
 {
 int last_dig=N%10;
 ans=ans+last_dig*pow;
 pow=pow*2;
 N=N/10;
 }
 cout<<ans<<endl;
 no=no-1;
 }
 return 0;
}

```

Problem Name: **Binary To Decimal**

Problem Difficulty: None

Problem Constraints:  $0 < N \leq 1000000000$

Problem Description:

Take N (number in binary format). Write a function that converts it to decimal format and Print the value returned.

Input Format:

Sample Input: 101010

Output Format:

Sample Output: 42

```
#include<iostream>
```

```

using namespace std;

int main() {

 unsigned long long int n;
 cin>>n;
 unsigned long long int sum=0;
 int pow=1;
 while(n!=0)
 {
 int last=n%10;
 sum=sum+last*pow;
 pow=pow*2;
 n=n/10;
 }
 cout<<sum;
 return 0;
}

```

Problem Name: **Decimal To Octal**

Problem Difficulty: None

Problem Constraints:  $0 < N \leq 1000000000$

Problem Description:

Take N (number in decimal format). Write a function that converts it to octal format. Print the value returned.

Input Format:

Sample Input: 63

Output Format:

Sample Output: 77

```

#include <iostream>
using namespace std;
int main() {
 int n;
 std::cin >> n;
}

```

```
string octal = "";
while(n!=0){
 int r = n%8;
 n/=8;
 char c = r+'0';
 octal = c+octal;
}
std::cout << octal << std::endl;
}
```

Problem Name: **Print Series**

Problem Difficulty: None

Problem Constraints:  $0 < N1 < 100$

$0 < N2 < 100$

Problem Description:

Take the following as input.

A number (N1)

A number (N2)

Write a function which prints first N1 terms of the series  $3n + 2$

which are not multiples of N2.

Input Format:

Sample Input: 10

4

Output Format:

Sample Output: 5

11

14

17

23

26

29

35

38

41

```
#include <iostream>
using namespace std;
int main() {
 int n1,n2;
 cin>>n1>>n2;
 int n=1;
 int count=1;
 while(count<=n1)
 {

 int ans=3*n+2;
 if(ans%n2!=0)
 {

 cout<<ans<<endl;
 count++;
 }
 n++;
 }

 for(int i=1;i<n1;i++);
}
```

Problem Name: Odd and Even back in Delhi

Problem Difficulty: 1

Problem Constraints:  $N \leq 1000$

Car No  $\geq 0$  && Car No  $\leq 1000000000$

Problem Description:

Due to an immense rise in Pollution, Kejriwal is back with the Odd and Even Rule in Delhi. The scheme is as follows, each car will be allowed to run on Sunday if the sum of digits which are even is divisible by 4 or sum of digits which are odd in that number is divisible by 3. However to check every car for the above criteria can't be done by the Delhi Police. You need to help Delhi Police by finding out if a car numbered N will be allowed to run on Sunday?

Input Format: The first line contains N , then N integers follow each denoting the number of the car.

Sample Input: 2

12345

12134

Output Format: N lines each denoting "Yes" or "No" depending upon whether that car will be allowed on Sunday or Not !

Sample Output: Yes

No

```
#include <iostream>

using namespace std;

int main ()
{
 unsigned long long int i;
 int n;
 cin >> n;
 unsigned long long int num[n];
 for (i = 1; i <= n; i++)
 cin >> num[i-1];

 for (i = 0; i < n; i++)
 {
 int sum_even = 0, sum_odd = 0;
 while (num[i] != 0)
```

```

{
 int t;
 t = num[i]%10;
 if (t%2 == 0)
 sum_even = sum_even + t;
 else if (t%2 == 1)
 sum_odd = sum_odd + t;

 num[i] = num[i]/10;
}
if (sum_even%4 == 0 || sum_odd%3 == 0)
 cout << "Yes" << endl;
else
 cout << "No" << endl;

}
return 0;
}

```

Problem Name: Conversion (Fahrenheit to Celsius)

Problem Difficulty: None

Problem Constraints:  $0 < \text{Min} < 100$

$\text{Min} < \text{Max} < 500$

$0 < \text{Step}$

Problem Description:

Take the following as input.

Minimum Fahrenheit value

Maximum Fahrenheit value

Step

Print as output the Celsius conversions. Use the formula  $C = (5/9)(F - 32)$  E.g. for an input of 0, 100 and 20 the output is

0 -17

20 -6

40 4

60 15

80 26

100 37

```
#include <iostream>
using namespace std;
int main ()
{
 int ll,hl,w;
 cin>>ll>>hl>>w;
 int i;
 int c;
 for(i=ll;i<=hl;i+=w)
 {
 c=(int)((5.0/9)*(i-32));
 cout<<i<<"\t"<<c<<endl;

 }

 return 0;
}
```

Input Format: The first line of the input contains an integer denoting the Minimum Fahrenheit value.

The second line of the input contains an integer denoting the Maximum Fahrenheit value.

The third line of the input contains an integer denoting the Step.

Sample Input: 0

100

20

Output Format: Print Fahrenheit and Celsius values separated by a tab. Each step should be printed in a new line.

Sample Output: 0 -17

20 -6

40 4

60 15

80 26

100 37

Problem Name: Check prime

Problem Difficulty: None

Problem Constraints:  $2 < N \leq 1000000000$

Problem Description:

Take as input a number N, print "Prime" if it is prime if not Print "Not Prime".

Input Format:

Sample Input: 3

Output Format:

Sample Output: Prime



```
#include <iostream>

using namespace std;

int main ()
{
 int n;
 cin>>n;

 int i;
 for(i=2;i<n-1;i++)
 {

 if(n%i==0)
 {

 cout<<"Not Prime"<<endl;
 break;
 }
 }

 if(i==n)
 {

 cout<<"Prime"<<endl;
 }

}
```

Problem Name: Count Digits

Problem Difficulty: None

Problem Constraints:  $0 \leq N \leq 1000000000$

$0 \leq \text{Digit} \leq 9$

Problem Description:

Take the following as input.

A number

A digit

Write a function that returns the number of times digit is found in the number. Print the value returned.

Input Format: Integer (A number)

Integer (A digit)

Sample Input: 5433231

3

Output Format: Integer (count of times digit occurs in the number)

Sample Output: 3

```
#include <iostream>
```

```
using namespace std;
```

```
int main ()
```

```
{
```

```
 int count=0;
```

```
 int n,digit;
```

```
 cin>>n>>digit;
```

```
 while(n!=0)
```

```

{

 if(digit==n%10)
 {
 count+=1;
 }
 n=n/10;
}
cout<<count;

}

```

Problem Name: Increasing Decreasing Sequence

Problem Difficulty: None

Problem Constraints:  $0 < N < 1000$

Each number in sequence  $S$  is  $> 0$  and  $< 1000000000$

Problem Description:

Given an array  $S$  of size  $N$ , check if it is possible to split sequence

into two sequences -

$s_{<sub>1</sub>}$  to  $s_{<sub>i</sub>}$  and  $s_{<sub>i+1</sub>}$  to  $s_{<sub>N</sub>}$

such that first sequence is strictly decreasing

and second is strictly increasing. Print true/false as output.

Input Format: First line contains a single integer  $N$  denoting the size

of the input. <br>

Next  $N$  lines contain a single integer each denoting the elements of the array  $S$ .

Sample Input: 5

1

2

3

4

5

Output Format: Print boolean output - "true" or "false" defining whether the sequence is increasing - decreasing or not.

Sample Output: true

=====Solution=====

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
 int n;
```

```
 cin >> n;
```

```
 int prev;
```

```
 cin >> prev;
```

```
 bool isValid = true;
```

```
 bool isDecreasing = true;
```

```
 while(--n) {
```

```
 int curr;
```

```
 cin >> curr;
```

```
 if(curr == prev) {
```

```
 isValid = false;
```

```
 break;
```

```
 }
```

```
 else if(curr > prev) {
```

```
 isDecreasing = false;
 }
 else if(!isDecreasing && curr < prev) {
 isValid = false;
 break;
 }

 prev = curr;
}

cout << boolalpha << isValid << endl;

return 0;
}
```

Problem Name: Is Armstrong Number

Problem Difficulty: None

Problem Constraints:  $0 < N < 1000000000$

Problem Description:

Take the following as input.

A number

Write a function which returns true if the number is an armstrong number and false otherwise, where Armstrong number is defined as follows.

A positive integer of n digits is called an Armstrong number of order n (order is number of digits) if.

$abcd... = \text{pow}(a,n) + \text{pow}(b,n) + \text{pow}(c,n) + \text{pow}(d,n) + ....$

1634 is an Armstrong number as  $1634 = 1^4 + 6^4 + 3^4 + 4^4$

371 is an Armstrong number as  $371 = 3^3 + 7^3 + 1^3$

Input Format: Single line input containing an integer

Sample Input: 371

Output Format: Print boolean output for each testcase. <br>

"true" if the given number is an Armstrong Number, else print "false".

Sample Output: true

=====Solution=====

```
#include <bits/stdc++.h>
```

```
using namespace std;
```

```
int fastPow(int a, int x)
```

```
{
```

```
 if (x == 0)
```

```
 {
```

```
 return 1;
```

```
 }
```

```
int ans = 1;
int k = 1;
while (k <= x)
{
 if (x & k)
 {
 ans = ans * a;
 }
 k <<= 1;
 a *= a;
}
return ans;
}
```

```
bool isArmstrong(int n)
{
 int noOfDigits = 0;
 int m = n;
 while (m)
 {
 noOfDigits++;
 m /= 10;
 }
```

```
 m = n;
 int sum = 0;
 while (m)
 {
 int r = m % 10;
```

```

 sum += fastPow(r, noOfDigits);
 m /= 10;
 }

 return n == sum;
}

int main()
{
 int n;
 cin >> n;

 cout << boolalpha << isArmstrong(n) << endl;

 return 0;
}

```

Problem Name: Basic Calculator

Problem Difficulty: None

Problem Constraints: 0 <= Input integers <= 1000000000 <br>

( It is assured that the second integer provided for division and modulo operations will not be 0. )

Problem Description:

Write a program that works as a simple calculator.

1.It reads a character (ch) <br>



- 2.If ch is among '+', '-', '\*', '/' or '%' it further takes two numbers (N1 and N2 as input). It then performs appropriate operation between numbers and print the number. <br>
- 3.If ch is 'X' or 'x', the program terminates. <br>
- 4.If ch is any other character, the program should print 'Invalid operation. Try again.' and seek inputs again (starting from character). <br><br>

Write code to achieve above functionality.

Input Format:

Sample Input: \*

1

2

/

4

2

+

3

2

;

X

Output Format: Output a single integer output for the operations in a new line each.

Sample Output: 2

2

5

Invalid operation. Try again.

=====Solution=====

```

import java.util.Scanner;

public class Main{

 static Scanner scn=new Scanner(System.in);

 public static void main(String[] args) {

 char ch;

 do {

 ch = scn.next().charAt(0); // Use cin in case of c++

 if (ch == '+' || ch == '-' || ch == '*' || ch == '/' || ch == '%') {

 operation(ch);

 } else {

 if (ch != 'x' && ch != 'X')

 System.out.println("Invalid operation. Try again.");

 }

 } while (ch != 'x' && ch != 'X');

 }

 public static void operation(char ch) {

 int a = scn.nextInt(); // Use cin in case of c++

 int b = scn.nextInt(); // Use cin in case of c++

 int res = 0;

 switch (ch) {

 case '+': {

 res = a + b;

```

```
 break;
 }
 case '-': {
 res = a - b;

 break;
 }
 case '*': {
 res = a * b;

 break;
 }
 case '/': {
 res = a / b;
 break;
 }
 case '%': {
 res = a % b;
 break;
 }
 }
 System.out.println(res);
}
```

Problem Name: Help Ramu

Problem Difficulty: 2

Problem Constraints:  $1 \leq T \leq 1000$  , where T is no of testcases <br>

$1 \leq c_1, c_2, c_3, c_4 \leq 1000$  <br>

$1 \leq n, m \leq 1000$  <br>

$0 \leq a_i, b_i \leq 1000$

Problem Description:

Ramu often uses public transport. The transport in the city is of two types: cabs and rickshaws. The city has n rickshaws and m cabs, the rickshaws are numbered by integers from 1 to n, the cabs are numbered by integers from 1 to m.

Public transport is not free. There are 4 types of tickets:

A ticket for one ride on some rickshaw or cab. It costs  $c_1$  rupees;

A ticket for an unlimited number of rides on some rickshaw or on some cab. It costs  $c_2$  rupees;

A ticket for an unlimited number of rides on all rickshaws or all cabs. It costs  $c_3$  rupees;

A ticket for an unlimited number of rides on all rickshaws and cabs. It costs  $c_4$  rupees.

Ramu knows for sure the number of rides he is going to make and the transport he is going to use. He asked you for help to find the minimum sum of rupees he will have to spend on the tickets.

Input Format: Each Test case has 4 lines which are as follows:

The first line contains four integers  $c_1, c_2, c_3, c_4$  ( $1 \leq c_1, c_2, c_3, c_4 \leq 1000$ ) — the costs of the tickets.  
<br>

The second line contains two integers  $n$  and  $m$  ( $1 \leq n, m \leq 1000$ ) — the number of rickshaws and cabs Ramu is going to use. <br>

The third line contains  $n$  integers  $a_i$  ( $0 \leq a_i \leq 1000$ ) — the number of times Ramu is going to use the rickshaw number  $i$ . <br>

The fourth line contains  $m$  integers  $b_i$  ( $0 \leq b_i \leq 1000$ ) — the number of times Ramu is going to use the cab number  $i$ .

Sample Input: 2

1 3 7 19

2 3

2 5

4 4 4

4 3 2 1

1 3

798

1 2 3

Output Format: For each testcase , print a single number - the minimum sum of rupees Ramu will have to spend on the tickets in a new line.

Sample Output: 12

1

====Solution====

//Rajkishor Ranjan

// Code Functionality.

#include <iostream>

#include <numeric>

using namespace std;

```
const int MAX = 1000;
```

```
int minFare(int c1 , int c2 , int c3 , int freq[] , int num)
```

```
{
```

```
 int cost = 0;
```

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

```
 {
```

```
 cost += min(c2 , freq[i] * c1);
```

```
 }
```

```
 int minFare = min(c3 , cost);
```

```
 return minFare;
```

```
}
```

```
int minSum(int c1 , int c2 , int c3 , int c4 , int rickshaws[] , int cabs[] , int n , int m)
```

```
{
```

```
 int rickshawFare = minFare(c1 , c2 , c3 , rickshaws , n);
```

```
 int cabFare = minFare(c1 , c2 , c3 , cabs , m);
```

```
 int minFare = min(rickshawFare + cabFare , c4);
```

```
 return minFare;
```

```
}
```

```
int main()
```

```
{
```

```
 int testCases;
```

```
cin >> testCases;

while(testCases > 0)
{
 int c1 , c2 , c3 ,c4;
 cin >> c1 >> c2 >> c3 >> c4;

 int n,m;
 cin >> n >> m;

 int rickshaws[MAX];
 int cabs[MAX];

 for(int i = 0 ; i < n ; i++)
 {
 cin >> rickshaws[i];
 }

 for(int i = 0 ; i < m ; i++)
 {
 cin >> cabs[i];
 }

 int result = minSum(c1 , c2 , c3 , c4 , rickshaws , cabs , n , m);

 cout << result;
 cout << "\n";

 testCases--;
```

```
}

return 0;

}
```

Problem Name: Help Ramu

Problem Difficulty: 2

Problem Constraints:  $1 \leq T \leq 1000$ , where T is no of testcases <br>

$a_i, b_i \leq 1000$

Problem Description:

Ramu often uses public transport. The transport in the city is of two types: cabs and rickshaws. The city has n rickshaws and m cabs, the rickshaws are numbered by integers from 1 to n, the cabs are numbered by integers from 1 to m.

Public transport is not free. There are 4 types of tickets:

A ticket for one ride on some rickshaw or cab. It costs c1 rupees;

A ticket for an unlimited number of rides on some rickshaw or on some cab.

It costs c2 rupees;

A ticket for an unlimited number of rides on all rickshaws or all cabs.

It costs c3 rupees;



A ticket for an unlimited number of rides on all rickshaws and cabs.

It costs  $c_4$  rupees.

Ramu knows for sure the number of rides he is going to make and the transport he

is going to use. He asked you for help to find the minimum sum of rupees he will have to spend on the tickets.

Input Format: Each Test case has 4 lines which are as follows:

The first line contains four integers  $c_1, c_2, c_3, c_4$  ( $1 \leq c_1, c_2, c_3, c_4 \leq 1000$ ) — the costs of the tickets.  
<br>

The second line contains two integers  $n$  and  $m$  ( $1 \leq n, m \leq 1000$ ) — the number of rickshaws and cabs Ramu is going to use. <br>

The third line contains  $n$  integers  $a_i$  ( $0 \leq a_i \leq 1000$ ) — the number of times Ramu is going to use the rickshaw number  $i$ . <br>

The fourth line contains  $m$  integers  $b_i$  ( $0 \leq b_i \leq 1000$ ) — the number of times Ramu is going to use the cab number  $i$ .

Sample Input: 2

1 3 7 19

2 3

2 5

4 4 4

4 3 2 1

1 3

798

1 2 3

Output Format: For each testcase , print a single number - the minimum sum of rupees Ramu will have to spend on the tickets in a new line.

Sample Output: 12

1

// Code Functionality.

Problem Name: Pythagoras Triplet

Problem Difficulty: None

Problem Constraints:  $N \leq 10^9$

Problem Description:

Given a number  $N$  (denoting one of the legs of the triangle), Print its Pythagoras pair in increasing order if they exist. Otherwise, print "-1".

Input Format: A single integer  $N$

Sample Input: 3

Output Format: Two numbers  $X$  and  $Y$  denoting the rest of the numbers of the Pythagorean triplet in increasing order.

Sample Output: 4 5

```

#include<cstdio>
#include<iostream>
#include<cmath>
using namespace std;
#define ll long long int
int main()
{
 ll x;
 while(cin >> x){
 if(x < 3){ printf("-1\n"); continue; }

 if(x & 1) printf("%lld %lld", (x*x-1)/2, (x*x+1)/2);
 else{

 printf("%lld %lld", ((x*x)/4)-1, ((x*x)/4)+1);
 }
 }
 return 0;
}

```

Problem Name: Fibonacci Pattern (Pattern 4)

Problem Difficulty: None

Problem Constraints:  $0 < N < 100$

Problem Description:

Take N (number of rows), print the following pattern (for N = 4)

```
0
1 1
2 3 5
8 13 21 34
```

Input Format:

Sample Input: 4

Output Format:

Sample Output: 0

```
1 1
2 3 5
8 13 21 34
```

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

int fib(int n)
{
 if(n==0)
 return 0;
 if(n==1)
 return 1;
 return fib(n-1)+fib(n-2);
}

int main()
{
 int n;
 cin>>n;
 int count =0;
 for(int i=0;i<=n;i++)
 {
 for(int j=0;j<i;j++)
 {
```

```
 cout<<fib(count)<<" ";
 count++;
 }
 cout<<endl;
}
}
```

Problem Name: Manmohan Loves Patterns - I

Problem Difficulty: 1

Problem Constraints:  $N \leq 1000$

Problem Description:

Given N, help Manmohan to print pattern upto N lines. For eg For N=6 , following pattern will be printed.

```
1
11
111
1001
11111
100001
```

Input Format: Single number N.

Sample Input: 6

Output Format: Pattern corresponding to N.

Sample Output: 1

```
11
111
1001
```

11111

100001

```
#include<iostream>
using namespace std;
int main()
{

 int n;
 cin>>n;
 int row=1;
 while(row<=n)
 {

 int col=1;
 if(row%2!=0)
 {

 //odd row
 while(col<=row)
 {

 cout<<1;
 col=col+1;
 }
 }
 }
```

```

else{
 //Even row
 cout<<1;
 while(col<=row-2)
 {

 cout<<0;
 col=col+1;
 }
 cout<<1;
}
cout<<endl;
row=row+1;
}
}

```

Given N, help Manmohan to print pattern upto N lines. For eg For N=6 , following pattern will be printed.

```

1
11
111
1001
11111
100001

```

Input Format

Single number N.

Constraints

$N \leq 1000$

Output Format

Pattern corresponding to N.

Sample Input

6

Sample Output

1

11

111

1001

11111

100001

Explanation

For every odd number row print 1, odd number of times and for every even number row , print first and last character as 1 and rest of middle characters as 0.

```
#include<iostream>
using namespace std;
int main()
{
 int n;
 cin>>n;
 int row=1;
 while(row<=n)
 {
 int col=1;
 if(row%2!=0)
 {
 //odd row
 while(col<=row)
 {
 cout<<1;
 col=col+1;
 }
 }
 row++;
 }
}
```



```

 }
 else{
 //Even row
 cout<<1;
 while(col<=row-2)
 {

 cout<<0;
 col=col+1;
 }
 cout<<1;
 }
 cout<<endl;
 row=row+1;
}
}

```

Problem Name: Manmohan Loves Patterns- II

Problem Difficulty: 1

Problem Constraints:  $N \leq 1000$

Problem Description:

Help Manmohan to print pattern of a given number. See the output pattern for given input  $n = 5$ .

Input Format: Single integer  $N$  denoting number of lines of the pattern.

Sample Input: 5

Output Format: Pattern.

Sample Output: 1

11

202

3003

40004

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
 int i, j, x = -1;
```

```
 int n;
```

```
 cin >> n;
```

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

```
 {
```

```
 if (x < 1)
```

```
 {
```

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

```
 cout << "1";
```

```
 }
```

```
 else
```

```
 {
```

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

```
 {
```

```
 if(j==1 || j==i)
```

```
 cout << i-1;
```

```
 else
```

```
 cout << "0";
```

```
 }
```

```
 }
```

```
 cout << endl;
```

```

 x++;
 }
 return 0;
}

```

Problem Name: Pattern Mountain

Problem Difficulty: None

Problem Constraints:  $0 < N < 10$

Problem Description:

Take N (number of rows), print the following pattern (for N = 4).

```

 1 1
 1 2 2 1
1 2 3 3 2 1
1 2 3 4 3 2 1

```

Input Format:

Sample Input: 4

Output Format:

Sample Output: 1

```

1 2 2 1
1 2 3 3 2 1
1 2 3 4 3 2 1

```

```
#include <iostream>
```

```
using namespace std;
```

```

int main ()
{
 int i, j, k = 1;
 int n;
 cin >> n;
 for (i = 1; i <= n; i++)
 {
 for (j = 1; j <= i; j++)
 {
 cout << k << " ";

 k++;
 }
 for (j = 1; j <= 2*n - 1 - 2*i; j++)
 cout << " ";
 for (j = 1; j <= i; j++)
 {
 k--;

 if (k == n)
 continue;

 cout << k << " ";
 }
 cout << endl;
 }
 return 0;
}

```

Problem Name: Pattern with Zeros

Problem Difficulty: None

Problem Constraints:  $0 < N < 100$

Problem Description:

Take N (number of rows), print the following pattern (for N = 5)

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

Input Format:

Sample Input: 5

Output Format:

Sample Output: 1

```
2 2
3 0 3
4 0 0 4
5 0 0 0 5
```

=====Solution=====

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
 int i, j;
```

```
 int n;
```

```
 cin >> n;
```

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

```

{

 for (j = 1; j <= i; j++)
 {
 if(j==1 || j==i)
 cout << i<<" ";

 else
 cout << "0"<<" ";

 }

 cout << endl;

}

return 0;
}

```

Problem Name: Pattern Triangle

Problem Difficulty: None

Problem Constraints:  $0 < N < 10$

Problem Description:

Take N (number of rows), print the following pattern (for N = 4).

```

 1
 2 3 2
3 4 5 4 3
4 5 6 7 6 5 4

```

Input Format:

Sample Input: 4

Output Format:

Sample Output:

```

 1
 2 3 2
 3 4 5 4 3
 4 5 6 7 6 5 4
```

=====Solution=====

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

```
public class patterntriangle {
```

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

```
 // TODO Auto-generated method stub
```

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

```
 int n=scn.nextInt();
```

```
 int nsp=n-1;
```

```
 int num=1;
```

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

```
 {
```

```
 //work for spaces
```

```
 for(int csp=1;csp<=nsp;csp++)
```

```

 {
 System.out.print(" \t");
 }

 // work for numbers

 for(int cst=1;cst<=(2*i-1);cst++)
 {
 System.out.print(num+"\t");

 if(cst<i)
 num++;
 else
 num--;

 }

 //preparation

 nsp=nsp-1;
 num+=2;
 System.out.println();

}

}

}

```



Problem Name: Pattern DoubleSidedArrow

Problem Difficulty: None

Problem Constraints: N is odd number.

Problem Description:

Take N as input. For a value of N=7, we wish to draw the following pattern :

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

Input Format: Take N as input.

Sample Input: 7

Output Format: Pattern should be printed with a space between every two values.

Sample Output:

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

=====Solution=====

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

```
/**
```

```
 * @author Garima Chhikara
```

```
 * @email garima.chhikara@codingblocks.com
```

```
 * @date 06-Apr-2018
```

```
 */
```

```
public class patternstar {
```

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

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

```
 int n = scn.nextInt() ;
```

```
 int nsp1 = n - 1;
```

```
 int nsp2 = -1;
```

```
 int nst = 1;
```

```
 for (int row = 1; row <= n; row++) {
```

```
 int val;
```

```
 if (row <= n / 2 + 1) {
```

```
 val = row;
```

```
 } else {
```

```
 val = n - row + 1;
```

```
 }
```

```
 for (int csp = 1; csp <= nsp1; csp++) {
```

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

```
 }
```

```
for (int cst = 1; cst <= nst; cst++) {
 System.out.print(val + " ");
 val-- ;
}
for (int csp = 1; csp <= nsp2; csp++) {
 System.out.print(" ");
}
```

```
int cst = 1;
val++ ;
if (row == 1 || row == n) {
 cst = 2;
```

```
}
for (; cst <= nst; cst++) {
 System.out.print(val + " ");
 val++ ;
}
```

```
if (row <= n / 2) {
 nsp1 -= 2;
 nst++;
 nsp2 += 2;
} else {
 nsp1 += 2;
 nst--;
 nsp2 -= 2;
}
```

```

 System.out.println();
 }
}

```

Problem Name: Pattern InvertedHourGlass

Problem Difficulty: None

Problem Constraints:

Problem Description:

Take N as input. For a value of N=5, we wish to draw the following pattern :

```

5 5
5 4 4 5
5 4 3 3 4 5
5 4 3 2 2 3 4 5
5 4 3 2 1 1 2 3 4 5
5 4 3 2 1 0 1 2 3 4 5
5 4 3 2 1 1 2 3 4 5
5 4 3 2 2 3 4 5
5 4 3 3 4 5
5 4 4 5
5 5

```

Input Format: Take N as input.

Sample Input: 5

Output Format: Pattern should be printed with a space between every two values.

Sample Output: 5 5

5 4 4 5

5 4 3 3 4 5

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

=====Solution=====

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

```
/**
```

```
 * @author Garima Chhikara
```

```
 * @email garima.chhikara@codingblocks.com
```

```
 * @date 05-Apr-2018
```

```
 */
```

```
public class patternhb2 {
```

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

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

```
 int n = scn.nextInt();
```

```

int rows = 2 * n + 1;

int nst = 1;
int nsp = 2 * n - 1;
for (int row = 1; row <= rows; row++) {
 int val = n;

 for (int cst = 1; cst <= nst; cst++) {
 System.out.print(val + " ");
 val--;
 }

 for (int csp = 1; csp <= nsp; csp++) {
 System.out.print(" ");
 }

 int cst = 1;
 if (row == n + 1) {
 cst = 2;
 val += 2;
 } else {
 cst = 1;
 val++;
 }

 for (; cst <= nst; cst++) {
 System.out.print(val + " ");
 val++;
 }
}

```

```

 if (row <= n) {
 nsp -= 2;
 nst++;
 } else {
 nsp += 2;
 nst--;
 }
 System.out.println();
 }
}

```

Problem Name: 卐 Ganesha's Pattern

Problem Difficulty: None

Problem Constraints:  $5 \leq N \leq 99$

Problem Description:

<img src = "https://image.flaticon.com/icons/png/512/805/805294.png" width="200px">

Take as input N, an odd number ( $\geq 5$ ) . Print the following pattern as given below for N = 7.

```

* ****
* *
* *

```

\* \*

\* \*

\*\*\*\* \*

Input Format: Enter value of N ( >=5 )

Sample Input: 7

Output Format: Print the required pattern.

Sample Output: \* \*\*\*\*

\* \*

\* \*

\*\*\*\*\*

\* \*

\* \*

\*\*\*\* \*

=====Solution=====

```
#include <iostream>
```

```
using namespace std;
```

```
int main ()
```

```
{
```

```
 int i, j;
```

```
 int n;
```

```
 cin >> n;
```

```
 cout << "*";
```

```
 for (i = 1; i <= (n+1)/2 - 2; i++)
```

```
 cout << " ";
```



```
for (i = 1; i <= (n+1)/2; i++)
```

```
 cout << "*";
```

```
cout << endl;
```

```
for (i = 1; i <= (n+1)/2 - 2; i++)
```

```
{
```

```
 cout << "*";
```

```
 for (j = 1; j <= (n+1)/2 - 2; j++)
```

```
 cout << " ";
```

```
 cout << "*";
```

```
 cout << endl;
```

```
}
```

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

```
 cout << "*";
```

```
cout << endl;
```

```
for (i = 1; i <= (n+1)/2 - 2; i++)
```

```
{
```

```
 for (j = 1; j <= (n+1)/2 - 1; j++)
```

```
 cout << " ";
```

```
 cout << "*";
```

```
 for (j = 1; j <= (n+1)/2 - 2; j++)
```

```

 cout << " ";

 cout << "*";

 cout << endl;
 }

 for (i = 1; i <= (n+1)/2; i++)
 cout << "*";

 for (i = 1; i <= (n+1)/2 - 2; i++)
 cout << " ";

 cout << "*";
 return 0;
}

```

Problem Name: Hollow Diamond Pattern(Pattern 6)

Problem Difficulty: None

Problem Constraints:  $0 < N < 10$  (where N is an odd number)

Problem Description:

Take N (number of rows), print the following pattern (for N = 5).

```

* * * * *
* * * *
* *
* * * *

```

\* \* \* \* \*

Input Format:

Sample Input: 5

Output Format:

Sample Output: \* \* \* \* \*

\* \* \* \*

\* \*

\* \* \*

\* \* \* \*

====Solution====

```
#include<iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
 int n,i=1,space=1;
```

```
 cin>>n;
```

```
 int cnt=n/2;
```

```
 if(n>0 && n <10 && n%2!=0)
```

```
 { while(i<=n/2+1)
```

```
 {
```

```
 if(i==1)
```

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

```
 cout<<"*\t";
```

```
 cout<<endl;
```

```
i=i+1;
```

```
}
```

```
else{
```

```
for(int j=1;j<=cnt;j++)
```

```
 cout<<"*\t";
```

```
for(int j=1;j<=space;j++)
```

```
 cout<<"\t";
```

```
space=space+2;
```

```
for(int j=1;j<=cnt;j++)
```

```
 cout<<"*\t";
```

```
 cout<<endl;
```

```
cnt=cnt-1;
```

```
i=i+1;
```

```
}
```

```
}
```

```
i=i-2;
```

```
cnt=cnt+2;
```

```
space=space-4;
```

```
while(i>=1)
```

```
{
```

```
if(i==1)
{ for(int j=1;j<=n;j++)
cout<<"*\t";
cout<<endl;
i=i-1;
}
```

```
else{
```

```
for(int j=1;j<=cnt;j++)
 cout<<"*\t";
```

```
for(int j=1;j<=space;j++)
 cout<<"\t";
space=space-2;
```

```
for(int j=1;j<=cnt;j++)
 cout<<"*\t";
cout<<endl;
```

```
cnt=cnt+1;
i=i-1;
}
```

```
 }
}
return 0;
}
```

Problem Name: Hollow Rhombus Pattern

Problem Difficulty: 1

Problem Constraints:  $N \leq 20$

Problem Description:

Given number of rows  $N$ , you have to print a Hollow Rhombus. See the output for corresponding given input.

Input Format: Single integer  $N$ .

Sample Input: 5

Output Format: Print pattern.

Sample Output: \*\*\*\*\*

```
* *

* *

* *

```

====Solution=====

```
/**
```

```
* @author ambika
```

```
*30-Oct-2018
```

```
*/
```

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

```

public class HollowRohmbusPattern {

 public static void main(String[] args) {

 Scanner scn = new Scanner(System.in);

 int n = scn.nextInt();

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

 // work for spaces
 for (int csp = 1; csp <= nsp; csp++) {
 System.out.print(" ");
 }

 // work for stars

 for (int cst = 1; cst <= nst; cst++) {
 System.out.print("*");
 }

 if (i > 1 && i < n) {
 for (int csp = 1; csp <= n - 2; csp++) {
 System.out.print(" ");
 }
 System.out.print("*");
 }
 }
 }
}

```

```
// preparation for next iteration
```

```
if (i >= 1 && i < n - 1) {
```

```
 nst = 1;
```

```
} else {
```

```
 nst = n;
```

```
}
```

```
nsp = nsp - 1;
```

```
System.out.println();
```

```
}
```

```
}
```

```
}
```



Problem Name: Pascal Triangle 1

Problem Difficulty: 1

Problem Constraints:  $N \leq 10$

Problem Description:

Given an integer N, print [Pascal Triangle]([https://en.wikipedia.org/wiki/Pascal's\\_triangle](https://en.wikipedia.org/wiki/Pascal's_triangle)) upto N rows.



Input Format: Single integer N.

Sample Input: 4

Output Format: Print pascal triangle.

Sample Output:        1

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

=====Solution=====

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

```
public class Main {
```

```
 public static void main(String[] args) {
 Scanner scn=new Scanner(System.in);
 // System.out.println("enter the no. N");
 int n=scn.nextInt(),i,j;
 for(i=1;i<=n;i++){
 int num=1;
 for(int spaces =1;spaces<=(n-i+1);spaces++)
 System.out.print(" ");
```

```

 for(j=1;j<=i;j++){
 if(j==1)
 System.out.print(j+" ");
 else
 {
 num=num*(i-j+1)/(j-1);
 System.out.print(num+" ");
 }
 }
 System.out.print("\n");
 }
}

```

Problem Name: Pattern Numbers & Stars - 1

Problem Difficulty: None

Problem Constraints:

Problem Description:

Take as input N, a number. Print the pattern as given in output section for corresponding input.

Input Format: Enter value of N

Sample Input: 5

Output Format: All numbers and stars are Space separated

Sample Output: 1 2 3 4 5

1 2 3 4 \*

1 2 3 \* \* \*

1 2 \* \* \* \* \*

1 \* \* \* \* \* \*

=====Solution=====

```
#include <iostream>
using namespace std;
int main ()
{
 int i, j, n;
 cin >> n;

 for (i = 1; i <= n; i++)
 {
 for (j = 1; j <= n+1-i; j++)
 cout << j << " ";
 for (j = 1; j <= 2*(i-1) - 1; j++)
 cout << "* ";
 cout << endl;
 }
 return 0;
}
```

Problem Name: Pattern Numbers & Stars - 2

Problem Difficulty: None

Problem Constraints:  $1 \leq N < 10$

Problem Description:

Take as input N, a number. Print the pattern as given in the input and output section.

Input Format: Enter value of N

Sample Input: 7

Output Format: Print the pattern.

Sample Output:

1\*\*\*\*\*

12\*\*\*\*\*

123\*\*\*\*\*

1234\*\*\*

12345\*\*

123456\*

1234567

=====Solution=====

```
#include <iostream>
```

```
using namespace std;
```

```
int main ()
```

```
{
```

```
 int i, j, n;
```

```
 cin >> n;
```

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

```
 {
```

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

```
 cout << j;
```

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

```
 cout << "*";
```

```
 cout << endl;
```

```
 }
```

```
 return 0;
```

```
}
```

Question print all prime

```
#include<iostream>
```

```
using namespace std;
```

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

```
{
```

```
 for(int i=2;i<n-1;i++)
```

```
 {
```

```
 if(n%i==0)
```

```
 return false;
```

```
 }
```

```
 return true;
```

```
}
```

```
void printPrime(int N)
```

```
{
```

```
 for(int i=2;i<=N;i++)
```

```
 {
```

```
 if(isPrime(i))
```

```
 {
```

```
 cout<<i<<" ";
```

```
 }
```

```
 }
```

```
}
```

```
int main()
```

```
{
 int n;
 cin>>n;
 printPrime(n);
}
```

Print Fibonacci series

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

int fib(int n)
{

 int a=0,b=1;
 int c;
 for(int i=1;i<=n-1;i++)
 {

 c=a+b;
 a=b;
 b=c;
 }
 return c;
}
int main()
```

```
{
 int n;
 cin>>n;
 cout<<fib(n);
}
```

Print ABC

AB

A

Pattern

```
#include<iostream>
using namespace std;
int main()
{
 int n;
 cin>>n;
 for(int i=1;i<=n;i++)
 {
 int cnt_alphabet=n-i+1;
 char alphabet='A';
 for(int j=1;j<=cnt_alphabet;j++)
 {
 cout<<alphabet;
 alphabet=alphabet+1;
 }
 }
}
```

```
 }
 cout<<endl;
}
```

```
}
```

Bubble sort

```
#include<iostream>
```

```
using namespace std;
```

```
void bubb(int a[],int n)
```

```
{
```

```
 //N-1 large element to the end
```

```
 for(int itr=1;itr<=n-1;itr++)
```

```
 {
```

```
 //Pairwise swapping inn the unsorted array
```

```
 for(int j=0;j<=(n-itr-1);j++)
```

```
 {
```

```
 if(a[j]>a[j+1])
```

```
 {
```

```
 swap(a[j],a[j+1]);
```

```
 }
```

```
 }
```

```
 }
```

```
}
```

```
int main()
```



```

{
 int n;
 cin>>n;
 int a[1000];
 for(int i=0;i<n;i++)
 {

 cin>>a[i];
 }
 bubb(a,n);
 for(int i=0;i<n;i++)
 {

 cout<<a[i]<<" ";
 }
}

```

Insertion Sort

```

#include<iostream>
using namespace std;
void insertion(int a[],int n)

{
 //N-1 large element to the end

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

```

```

 int e=a[i];
 //place the current element into the right
 int j=i-1;
 while(j>=0 and a[j]>e)
 {

 a[j+1]=a[j];
 j=j-1;
 }
 a[j+1]=e;
}
}
int main()
{
 int n;
 cin>>n;
 int a[1000];
 for(int i=0;i<n;i++)
 {

 cin>>a[i];
 }
 insertion(a,n);
 for(int i=0;i<n;i++)
 {

 cout<<a[i]<<" ";
 }
}

```

STL function

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

//define a comparator
bool compare(int a,int b)
{
 cout<<"comparing "<<a<<"and"<<b<<endl;
 return a<b;
}

int main()
{

 int n;
 cin>>n;
 int a[1000];
 for(int i=0;i<n;i++)
 {

 cin>>a[i];

 }

 sort(a,a+n,compare);
```

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

 cout<<a[i]<<" ";

}
}
```

SUrray

```
#include<iostream>
using namespace std;
int main()
{

 int n;

 cin>>n;

 int a[1000];

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

 cin>>a[i];

 }

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

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

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

 cout<<a[k]<<" ";

 }
 cout<<endl;
 }
}
return 0;
}
```

maxSubArray

```
#include<iostream>
```

```
using namespace std;
```

```
void maxSub(int a[],int n)
```

```
{
```

```
 int currentSum=0;
```

```
 int maxSum=0;
```

```
 int left=-1;
```

```
 int right=-1;
```

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

```
 {
```

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

```
 {
```

```

 currentSum=0;
 for(int k=i;k<=j;k++)
 {
 currentSum+=a[k];

 }
 if(currentSum>maxSum)
 {

 maxSum=currentSum;
 left=i;
 right=j;
 }
 }
}
//print the maxisum
cout<<"maximum sum is :"<<maxSum<<endl;
//print the subarray

for(int k=left;k<=right;k++)
{

 cout<<a[k]<<" ";
}

}

int main()

```

```

{

 int n;
 cin>>n;
 int a[1000];
 for(int i=0;i<n;i++)
 {

 cin>>a[i];

 }
 maxSub(a,n);

}

```

Optimized code

```

#include<iostream>
using namespace std;

void maxSub(int a[],int n)
{
 int cumSum[1000]={0};
 int currentSum=0;
 int maxSum=0;

 int left=-1;

```

```

int right=-1;
cin>>a[0];
cumSum[0]=a[0];
for(int i=1;i<n;i++)
{

 cin>>a[i];
 cumSum[i]=cumSum[i-1]+a[i];
}

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

 for(int j=i;j<n;j++)
 {
 currentSum=0;
 currentSum=cumSum[j]-cumSum[i-1];
 //update maximum is cs>maximumSum
 if(currentSum>maxSum)
 {

 maxSum=currentSum;
 left=i;
 right=j;
 }
 }
}

//print the maxisum

```



```

cout<<"maximum sum is : "<<maxSum<<endl;
//print the subarray

for(int k=left;k<=right;k++)
{

 cout<<a[k]<<" ";

}

}
int main()
{

 int n;
 cin>>n;
 int a[1000];

 maxSub(a,n);

}

```

Using Kadanes Algorithm

```

#include<iostream>
using namespace std;
int main()
{

```

```

int n;

cin>>n;

int a[1000];

int currentSum=0;

int maxSum=0;

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

 cin>>a[i];
}

//use kadane's algorithm
for(int i=0;i<n;i++)
{

 currentSum=currentSum+a[i];
 if(currentSum<0)
 {
 currentSum=0;
 }

 maxSum=max(currentSum,maxSum);
}

cout<<"maximum is "<<maxSum<<" ";
}

```

Pair Sum Using Two Pointer Approach

```
#include<iostream>
```

```
using namespace std;

int main()
{

 int a[]={1,3,5,7,10,11,12,13};
 int sum=16;
 int i=0;
 int j=sizeof(a)/sizeof(int)-1;
 while(i<j)
 {
 int currentSum=a[i]+a[j];
 if(currentSum>sum)
 {
 j--;
 }
 else if(currentSum<sum)
 {
 i++;
 }
 else if(currentSum==sum)
 {
 cout<<a[i]<<" and "<<a[j]<<endl;
 i++;
 j--;
 }
 }
}
```

Check if it is Palindrome or not

```
#include<iostream>
```

```
#include<cstring>
```

```
using namespace std;
```

```
bool isPalindrome(char a[])
```

```
{
```

```
 int i=0;
```

```
 int j=strlen(a)-1;
```

```
 while(i<j)
```

```
 {
```

```
 if(a[i]==a[j])
```

```
 {
```

```
 i++;
```

```
 j--;
```

```
 }
```

```
 else
```

```
 {
```

```
 return false;
```

```
 }
```

```
}
```

```
 return true;
```

```
}
```

```
int main()
{
 char a[1000];
 cin.getline(a,1000);
 if(isPalindrome(a))
 {

 cout<<"is Palindrome";
 }
 else
 {
 cout<<"is not Palindrome";

 }

}
```

Check string is Palindromic or not

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

void removeDuplicate(char a[])
{
```

```

int l=strlen(a);
if(l==1 or l==0)
{

 return;
}
int prev=0;
for(int current=0;current<l;current++)
{

 if(a[current]!=a[prev])
 {

 prev++;

 a[prev]=a[current];
 }
}
a[prev+1]='\0';
return;
}
int main()
{
 char a[1000];
 cin.getline(a,1000);
 removeDuplicate(a);
 cout<<a;

}

```

## Largest string

```
#include<iostream>
#include<cstring>
using namespace std;
int main()
{

 int n;
 cin>>n;
 char a[1000];
 char largest[1000];
 int len=0;
 int largest_len=0;
 cin.get();
 for(int i=0;i<n;i++)
 {

 cin.getline(a,1000);
 len=strlen(a);
 if(len>largest_len)
 {
 largest_len=len;
 strcpy(largest,a);

 }
 }
}
```

```
cout<<largest<<"ans"<<largest_len<<endl;
}
```



2D array

```
#include<iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
 int a[5][3]={0};
```

```
 //Iterate over the row
```

```
 int val=1;
```

```
 for(int row=0;row<=4;row++)
```

```
 {
```

```
 for(int col=0;col<=2;col++)
```

```
 {
```

```
 a[row][col]=val;
```

```
 val=val+1;
```

```
 cout<<a[row][col]<<" ";
```

```
 }
```

```
 cout<<endl;
```

```
 }
```

```
}
```

Print Like A wave

```
#include<iostream>
```

```
using namespace std;

int main()
{

 int a[5][3]={0};

 //Iterate over the row
 int val=1;
 for(int row=0;row<=4;row++)
 {

 for(int col=0;col<=2;col++)
 {
 a[row][col]=val;
 val=val+1;
 cout<<a[row][col]<<" ";
 }
 cout<<endl;
 }

 for(int col=0;col<3;col++)
 {

 if(col%2==0)
 {
 //top -down
 for(int row=0;row<5;row++)
 {
```

```

 cout<<a[row][col]<<" ";
 }
}
else
{
 //bottom -up
 for(int row=3;row>=0;row--)
 {
 cout<<a[row][col]<<" ";
 }
}
}
}

```

### Spiral Pattern

```
#include<bits/stdc++.h>
```

```
using namespace std;
```

```
void spiralprint(int a[5][4],int m,int n)
```

```
{
```

```
 int startRow=0;
```

```
 int startCol=0;
```

```
 int endRow=m-1;
```

```
 int endCol=n-1;
```

```
 while(startRow<=endRow && startCol<=endCol)
```

```

{
 //first Row
 for(int i=startCol;i<=endCol;i++)
 {
 cout<<a[startRow][i]<<" ";
 }
 startRow++;

 //end column
 for(int i=startRow;i<=endRow;i++)
 {
 cout<<a[i][endCol]<<" ";
 }
 endCol--;

 //Bottom Row
 if(endRow>startRow)
 {

 for(int i=endCol;i>=startCol;i--)
 {
 cout<<a[endRow][i]<<" ";
 }
 endRow--;
 }

 //start col
 if(endCol>startCol)
 {

```

```
 for(int i=endRow;i>=startRow;i--)
 {

 cout<<a[i][startCol]<<" ";

 }
 startCol++;
 }
}
```

```
 }
}

int main()
{
 int a[5][4]={0};
 int m,n;
 cin >> m >> n;
 //Iterate over the array
 int val=1;
 for(int row=0;row<=m-1;row++)
 {
 for(int col=0;col<=n-1;col++)
 {
 a[row][col]=val;
 val=val+1;
 cout<<a[row][col]<<" ";

 }
 cout<<endl;
 }
}
```

```
spiralprint(a,m,n);
```

```
}
```

Piyush is lost in a magical park which contains N rows and M columns. In order to get out of park safely and return home, he needs atleast K amount of strength. Given a N by M pattern, your task is to find whether Piyush can ever escape the park.

Piyush enters the park with strength S. The park is filled with some obstacles denoted by '.', some magical beans denoted by '\*' and some blockades denoted as '#'. If he encounters an obstacle (.), strength decreases by 2. If he encounters a magic bean ('\*'), his strength increases by 5. Piyush can only walk row wise, so he starts from left of a row and moves towards right and he does this for every row. However when he encounters a blockade (#), he cannot go any further in his current row and simply jumps to the start of a new line without losing any strength. Piyush requires a strength of 1 for every step. His strength should always be greater than K while traversing or else Piyush will get lost. Assume that Piyush can shift immediately from last of one row to the start of next one without loss of any strength, help out Piyush to escape the park. His escape is successful if he is able to return home with atleast K strength.

Input Format

First line of input contains four integers – N,M,K and S. Next N lines contain M space separated characters which can be '.', '\*' or '#'.

Constraints

$1 \leq N, M, K, S \leq 100$

Output Format

Print "Yes" or "No" depending on whether Piyush can escape or not. If the answer is "Yes", also print the amount of strength he escaped with.

Sample Input

```
4 4 5 20
. . * .
. # . .
* * . .
. # * *
```

## Sample Output

Yes

13

## Explanation

Piyush starts with strength  $S=20$ .

For first row, he encounters an obstacle '.' and his strength reduces by 3 (  $2+1$  ( 1 for taking the step) ). Similarly after the second obstacle, his strength reduces by 3 again and becomes  $S=14$ . Then he encounters a '\*', and his strength increases by 5 but decreases by 1 for taking the step. Then his strength reduces by 2 (Not 3 as he will jump with no extra strength from here) after the last '.'. At the end of the first row his strength is  $S=16$ .

In the second row, he encounters a '.' and his strength reduces by 3 (  $2+1$  for the '.' ). Then he encounters a '#' and without losing any extra strength simply jumps to the first cell of the next row. Similarly, his strength at the beginning of the third row is 13 and after completing it, his strength is 16.

In the fourth row, he first encounters a '.' and his strength reduces to 13. Then he encounters a '#' at the second position and jumps to the next row. Since this is the last row, when he jumps he escapes from the park.

His strength left is 13. Since this is clearly greater than  $K=5$ , his escape was successful.

Piyush escaped with final strength = 13.

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

void magical_park(char a[][100],int m,int n,int k,int s)
{
 //piyush can get out to the mark
 bool success=true;

 for(int i=0;i<m;i++)
 {
 for(int j=0;j<n;j++)
 {
 char ch=a[i][j];
 //check
 if(s<k)
 {
 success=false;
 break;
 }
 if(ch=='*')
 {

```

```

 s+=5;

 }
 else if(ch=='.')
 {
 s-=2;
 }
 else
 {
 break;
 }

 //we also loose 1 point when we move right except last column
 if(j!=n-1)
 {
 s--;
 }
}
}
if(success)
{
 cout<<"Yes"<<endl;
 cout<<s<<endl;
}
else
{
 cout<<"No"<<endl;
}

}

int main()
{

 int m,n,k,s;
 cin>>m>>n>>k>>s;
 char park[100][100];

 //Take input
 for(int i=0;i<m;i++)
 {

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

```



```
{

 cin>>park[i][j];
}
}
magical_park(park,m,n,k,s);
return 0;
}
```

Given a 2D array of size  $N \times N$ . Rotate the array 90 degrees anti-clockwise.



Input Format

First line contains a single integer N. Next N lines contain N space separated integers.

Constraints

$N < 1000$

Output Format

Print N lines with N space separated integers of the rotated array.

Sample Input

```
4
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
```

Sample Output

```
4 8 12 16
3 7 11 15
2 6 10 14
1 5 9 13
```

Explanation

Rotate the array 90 degrees anticlockwise.

```
#include<bits/stdc++.h>
using namespace std;
void display(int arr[][100],int n)
{
 for(int i=0;i<n;i++)
 {
 for(int j=0;j<n;j++)
 {
 cout<<arr[i][j];
 }
 cout<<endl;
 }
}

void rotateimage(int arr[][100],int n)
{
 //reverse each row
 for(int row=0;row<n;row++)
 {
```

```

 int start_col=0;
 int end_col=n-1;
 while(start_col<end_col)
 {

 swap(arr[row][start_col],arr[row][end_col]); //reverse each row
 start_col++;
 end_col--;

 }
}
//void rotateimage_stl(int arr[][100],int n)
//take transpose of the matrix

for(int i=0;i<n;i++)
{
 for(int j=0;j<n;j++)
 {
 if(i<j){
 swap(arr[i][j],arr[j][i]);
 }
 }
}
}
int main()
{

 int n;
 cin>>n;
 int arr[100][100];
 for(int i=0;i<n;i++)
 {

 for(int j=0;j<n;j++)
 {
 cin>>arr[i][j];
 }
 }
 rotateimage(arr,n);
 display(arr,n);
 return 0;
}

```



Search in 2D array

```
#include<bits/stdc++.h>
```

```
using namespace std;
```

```
void searchArray(int arr[][100],int n,int element)
```

```
{
```

```
 int start_row=0;
```

```
 int end_col=n-1;
```

```
 while(start_row<n && end_col>=0)
```

```
 {
```

```
 if(arr[start_row][end_col]==element)
```

```
 {
```

```
 cout<<"element found at"<<start_row<<","<<end_col<<" ";
```

```
 }
```

```
 if(arr[start_row][end_col]>element)
```

```
 {
```

```
 end_col--;
```

```
 }
```

```
else
```

```
{
```

```
 start_row++;
```

```
}
```

```
}
```

```
}
```

```
int main()
```

```
{
```

```
 int n,element;
```

```
 cin>>n>>element;
```

```
 int arr[100][100];
```

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

```
 {
```

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

```
 {
```

```
 cin>>arr[i][j];
```

```
 }
```

```
 }
```

```
 searchArray(arr,n,element);
```

```
 return 0;
}
```

1.It reads a number N.

2.Take Another N numbers as input and store them in an Array.

3.calculate the max value in the array and return that value.

Input Format: First line contains integer n as size of array. Next n lines contains a single integer as element of array.

Sample Input: 4

2

8

6

4

Output Format: Print the required output.

Sample Output: 8

```
#include<bits/stdc++.h>
```

```
using namespace std;
```

```
void findmax(long long int arr[],long long int n)
```

```
{
```

```
 int max=-1000000000;
```

```
 long long int i;
```

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

```

{

 if(max<arr[i])
 {

 max=arr[i];
 }
}
cout<<max<<endl;
}
int main()
{
 long long int n;
 cin>>n;
 long long int a[100];
 long long int i;
 for(i=0;i<n;i++)
 {

 cin>>a[i];
 }
 findmax(a,n);

}

```

Problem Name: Arrays-Max Value In Array

Problem Difficulty: None

Problem Constraints: N cannot be Negative. Range of Numbers can be between -1000000000 to 1000000000

Problem Description:

Take as input N, the size of array. Take N more inputs and store that in an array. Write a function which returns the maximum value in the array. Print the value returned.

Problem Name: Arrays-Wave print Column Wise

Problem Difficulty: None

Problem Constraints: Both M and N are between 1 to 10.

Problem Description:

Take as input a two-d array. Wave print it column-wise.

Input Format: Two integers M(row) and N(column) and further M \* N integers(2-d array numbers).

Sample Input: 4 4

11 12 13 14

21 22 23 24

31 32 33 34

41 42 43 44

Output Format: All M \* N integers separated by commas with 'END' written in the end(as shown in example).

Sample Output: 11, 21, 31, 41, 42, 32, 22, 12, 13, 23, 33, 43, 44, 34, 24, 14, END

```
#include<bits/stdc++.h>
```

```
using namespace std;
```

```
void printWave(long long int arr[][100],long long int m,long long int n)
```

```
{
```

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

```
 {
```

```
 if(j%2==0)
```

```
 {
```

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



```

 {
 cout<<arr[i][j]<<" ";
 }

 }
 else
 {

 for(int i=m-1;i>=0;i--)
 {
 cout<<arr[i][j]<<" ";
 }

 }

}

cout<<"END";
}

int main()

{
 long long int m,n;
 cin>>m>>n;
 long long int a[100][100];
 for(int i=0;i<m;i++)
 {

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

```

```

 cin>>a[i][j];
 }
}
printWave(a,m,n);
}

```

Problem Name: Arrays-Target Sum Pairs

Problem Difficulty: None

Problem Constraints: Length of the arrays should be between 1 and 1000.

Problem Description:

Take as input N, the size of array. Take N more inputs and store that in an array. Take as input “target”, a number. Write a function which prints all pairs of numbers which sum to target.

Input Format: The first line contains input N. Next N lines contains the elements of array and (N+1)th line contains target number.

Sample Input: 5

1

3

4

2

5

5

Output Format: Print all the pairs of numbers which sum to target. Print each pair in increasing order.

Sample Output: 1 and 4

2 and 3

```
#include<bits/stdc++.h>
```

```
using namespace std;
```

```
void findPairSum(int arr[],int n,int element)
{
 sort(arr,arr+n);

 int left=0;
 int right=n-1;
 while(left<right)
 {

 if(arr[left]+arr[right]==element)
 {

 cout<<arr[left]<<"and"<<arr[right]<<endl;
 left++;
 right--;
 }
 else if(arr[left]+arr[right]<element)
 {
 left++;
 }
 else
 {
 right--;
 }
 }
}

int main()
{
```

```
int n, element;
cin>>n>>element;
int a[1000];
for(int i=0;i<n;i++)
{

 cin>>a[i];
}
findPairSum(a,n,element);
}
```

Problem Name: Arrays-Target Sum Triplets

Problem Difficulty: None

Problem Constraints: Length of Array should be between 1 and 1000.

Problem Description:

Take as input N, the size of array. Take N more inputs and store that in an array. Take as input “target”, a number. Write a function which prints all triplets of numbers which sum to target.

Input Format: First line contains input N. <br> Next line contains N space separated integers denoting the elements of the array. <br> The third line contains a single integer T denoting the target element.

Sample Input: 9

5 7 9 1 2 4 6 8 3

10

Output Format: Print all the triplet present in the array in a new line each. The triplets must be printed as A, B and C where A,B and C are the elements of the triplet (  $A \leq B \leq C$ ) and all triplets must be printed in sorted order. Print only unique triplets.

Sample Output: 1, 2 and 7

1, 3 and 6

1, 4 and 5

2, 3 and 5

```
#include<bits/stdc++.h>

using namespace std;

void findPairSum(int arr[],int n,int element)
{
 sort(arr,arr+n);

 for (int i = 0; i < n; i++)
 {
 int left = i + 1;

 int right=n-1;
 while(left<right)
 {

 if(arr[i]+arr[left]+arr[right]==element)
 {

 cout<<arr[i]<<" "<<arr[left]<<"and"<<arr[right]<<endl;
 left++;
 right--;
```

```

 }
 else if(arr[i]+arr[left]+arr[right]<element)
 {
 left++;
 }
 else
 {
 right--;
 }
}
}
}
int main()
{

 int n, element;
 cin>>n>>element;
 int a[1000];
 for(int i=0;i<n;i++)
 {

 cin>>a[i];
 }
 findPairSum(a,n,element);
}

```

Problem Name: Rain Water Harvesting

Problem Difficulty: None

Problem Constraints:  $1 \leq N \leq 10^6$

### Problem Description:

Apoorvaa has created an elevated roof. She wants to know how much water can she save during rain.

Given n non negative integers representing the elevation map where width of every bar is 1, Find the maximum water that she can save.

Explanation for the Sample input Testcase:



So the total units of water she can save is 5 units

Input Format: First line contains an integer n. Second line contains n space separated integers representing the elevation map.

Sample Input: 10

0 2 1 3 0 1 2 1 2 1

Output Format: Print a single integer containing the maximum unit of waters she can save.

Sample Output: 5

```
#include<bits/stdc++.h>

using namespace std;

int main()
{
 int n;

 cin>>n;

 int left[1000],right[1000];

 int a[1000];

 for(int i=0;i<n;i++) //Take Input
 { cin>>a[i]; }

 left[0]=a[0]; //intialize left==0

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

```

{

 left[i]=max(left[i-1],a[i]); //ab ckeck karenge left le aage aur pichhe kaun bada hai
}
right[n-1]=a[n-1]; //right ko right side se start karenge
for(int i=n-2;i>=0;i--)
{
 right[i]=max(right[i+1],a[i]); //yha bhi check karenge bada kaun hai
}
int ans=0;
for(int i=0;i<n;i++)
{
 ans+=min(left[i],right[i])-a[i]; //ab left aur right me jo chhota hoga utana hi pani store karega
}

cout<<ans;

}

```

Problem Name: Maximum Subarray Sum

Problem Difficulty: None

Problem Constraints:  $1 \leq N \leq 100000$

$1 \leq t \leq 20$

$-100000000 \leq A[i] \leq 100000000$

Problem Description:



You are given a one dimensional array that may contain both positive and negative integers, find the sum of contiguous subarray of numbers which has the largest sum. For example, if the given array is {-2, -5, 6, -2, -3, 1, 5, -6}, then the maximum subarray sum is 7.

Input Format: The first line consists of number of test cases T. Each test case consists of two lines. <br>

The first line of each testcase contains a single integer N denoting the number of elements for the array A. <br>

The second line of each testcase contains N space separated integers denoting the elements of the array.

Sample Input: 2

4

1 2 3 4

3

-10 5 10

Output Format: Output a single integer denoting the maximum subarray sum for each testcase in a new line.

Sample Output: 10

15

```
#include<bits/stdc++.h>
```

```
using namespace std;
```

```
void maxSum(int a[1000],int n)
```

```
{
```

```
 int currentsum=0;
```

```
 int maxSum=0;
```

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

```
 {
```

```
 currentsum+=a[i];
```

```

 if(currentsum<0)
 currentsum=0;
 maxSum=max(maxSum,currentsum);

 }
 cout<<maxSum;
}

int main()
{
 int n;
 cin>>n;
 int a[1000];
 for(int i=0;i<n;i++)
 {

 cin>>a[i];
 }
 maxSum(a,n);

}

```

Problem Name: Maximum Circular Sum

Problem Difficulty: None

Problem Constraints:  $1 \leq t \leq 100$  <br>

$1 \leq n \leq 1000$  <br>

$|A_{\text{i}}| \leq 10000$

Problem Description:

You are provided n numbers (both +ve and -ve). Numbers are arranged in a circular form. You need to find the maximum sum of consecutive numbers.

Input Format: Fi~~~~~1st line contains integer t which is number of test case.<br>

For each test case, it contains an integer n which is the size of array and next line contains n space separated integers denoting the elements of the array.

Sample Input: 1

7

8 -8 9 -9 10 -11 12

Output Format: Print the maximum circular sum for each testcase in a new line.

Sample Output: 22

```
#include<bits/stdc++.h>
```

```
using namespace std;
```

```
int kadanes(int arr[1000],int n)
```

```
{
```

```
 if(n==0)
```

```
 return 0;
```

```
 int cs=0;
```

```
 int ms=INT_MIN;
```

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

```
 {
```

```
 cs+=arr[i];
```

```
 ms=max(ms,cs);
 if(cs<0)
 cs = 0;
}
return ms;
}
```

```
int maxCircular(int a[1000],int n)
{
```

```
 int maxkadanes=kadanes(a,n);
 if(maxkadanes<0)
 return maxkadanes;
 int arraySummaxWrap=0;
 for(int i=0;i<n;i++)
 {
```

```
 arraySummaxWrap+=a[i];
 a[i]=-a[i];
 }
```

```
 int nonValue=kadanes(a,n);
```

```

 int originalSum=arraySummaxWrap+nonValue;
 return max(originalSum,nonValue);
}
```

```
int main()
{
```

```

int n;

cin>>n;

int a[1000];

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

{

 cin>>a[i];

}

cout<<maxCircular(a,n);

}

```

Problem Name: Maximum length Biotonic Subarray

Problem Difficulty: None

Problem Constraints:  $1 \leq t \leq 100$

$1 \leq n \leq 1000000$

Problem Description:

You are provided n numbers of array. You need to find the maximum length of bitonic subarray. A subarray  $A[i \dots j]$  is biotonic if there is a k with  $i \leq k \leq j$  such that  $A[i] \leq A[i + 1] \dots \leq A[k] \geq A[k + 1] \geq \dots \geq A[j - 1] \geq A[j]$  i.e subarray is first increasing and then decreasing or entirely increasing or decreasing.

Input Format: First line contains integer t which is number of test case.

For each test case, it contains an integer n which is the size of array and next line contains n space separated integers.

Sample Input: 2

6

12 4 78 90 45 23

4

40 30 20 10

Output Format: Print the maximum length.

Sample Output: 5

4

```
#include<bits/stdc++.h>
```

```
using namespace std;
```

```
int maxLength(int arr[1000],int n)
```

```
{
```

```
 int inc[n];
```

```
 int dec[n];
```

```
 int max;
```

```
 inc[0]=1;
```

```
 dec[n-1]=1;
```

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

```
 inc[i]=(arr[i]>=arr[i-1])? inc[i-1]+1:1;
```

```
 for(int i=n-2;i>=0;i--)
```

```
 dec[i]=(arr[i]>=arr[i+1])? dec[i+1]+1:1;
```

```
 max=inc[0]+dec[0]-1;
```

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

```
 if(inc[i]+dec[i]-1>max)
```

```
 max=inc[i]+dec[i]-1;
```

```
 return max;
```

```

}
int main()
{

 int t;
 scanf("%d", &t);
 for(int i=0;i<t;i++)
 {
 int n;
 scanf("%d", &n);
 int a[n];
 for(int j=0;j<n;j++)
 {
 scanf("%d", &a[j]);
 }
 printf("%d \n",
 maxLength(a, n));
 }

 return 0;
}

```

Problem Name: Arrays-Spiral Print Anticlockwise

Problem Difficulty: None

Problem Constraints: Both M and N are between 1 to 10.

Problem Description:

Take as input a 2-d array. Print the 2-D array in spiral form anti-clockwise.

Input Format: Two integers M(row) and N(column) and further M \* N integers(2-d array numbers).

Sample Input: 4 4

11 12 13 14

21 22 23 24

31 32 33 34

41 42 43 44

Output Format: All M \* N integers separated by commas with 'END' written in the end(as shown in example).

Sample Output: 11, 21, 31, 41, 42, 43, 44, 34, 24, 14, 13, 12, 22, 32, 33, 23, END

=====Solution=====

```
#include <iostream>
```

```
using namespace std;
```

```
void input2D(int mat[][10], int m, int n){
```

```
 for(int i = 0; i < m; ++i){
```

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

```
 cin >> mat[i][j];
```

```
 }
```

```
 }
```

```
}
```

```
void printSpiral(int mat[][10], int m, int n){
```

```
 int left = 0;
```

```
 int right = n - 1;
```

```
 int top = 0;
```

```
 int bottom = m - 1;
```

```
 while(left <= right || top <= bottom){
```



```

if (left <= right){

 //print top to bottom : right elements
 for(int i = top; i <= bottom; ++i){
 cout << mat[i][left] << " ";
 }
 left = left + 1;
}

//print left to right : top elements
if(top <= bottom){
 for(int i = left; i <= right; ++i){
 cout << mat[bottom][i] << " ";
 }
 bottom = bottom - 1;
}

//print bottom to top : left
if (left <= right){
 for(int row = bottom; row >= top; --row){
 cout << mat[row][right] << " ";
 }
 right--;
}

//print right to left : bottom
if (top <= bottom){
 for(int col = right; col >= left; --col){
 cout << mat[top][col] << " ";
 }
}

```

```

 }
 ++top;
 }

}

cout << "END";
}

```

```

int main(){
 int mat[10][10];
 int m, n;
 cin >> m >> n;
 input2D(mat, m, n);

 printSpiral(mat, m, n);
}

```

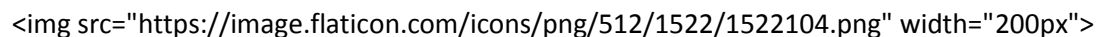
Problem Name: Rotate Image (N X N Array)

Problem Difficulty: None

Problem Constraints:  $N < 1000$

Problem Description:

Given a 2D array of size  $N \times N$ . Rotate the array 90 degrees anti-clockwise.

Input Format: First line contains a single integer  $N$ . Next  $N$  lines contain  $N$  space separated integers.

Sample Input: 4

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 16

Output Format: Print N lines with N space separated integers of the rotated array.

Sample Output: 4 8 12 16

3 7 11 15

2 6 10 14

1 5 9 13

=====Solution=====

```
#include <iostream>
```

```
#include<algorithm>
```

```
using namespace std;
```

```
void rotate(int a[][1000], int n){
```

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

```
 reverse(a[i],a[i]+n);
```

```
 }
```

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

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

```
 if(i<j){
```

```
 swap(a[i][j],a[j][i]);
```

```
 }
```

```
 }
```

```
 }
```

```
}
```

```
int main() {
```

```
 int n;
```

```
 cin>>n;
```

```

int a[1000][1000];
for(int i=0;i<n;i++){
 for(int j=0;j<n;j++){
 cin>>a[i][j];
 }
}
rotate(a,n);
for(int i=0;i<n;i++){
 for(int j=0;j<n;j++){
 cout<<a[i][j]<<" ";
 }
 cout<<endl;
}
}

```

Problem Name: Form Biggest Number

Problem Difficulty: None

Problem Constraints:  $1 \leq t \leq 100$  <br>

$1 \leq m \leq 100$  <br>

$1 \leq A[i] \leq 10^5$

Problem Description:

You are provided an array of numbers. You need to arrange them in a way that yields the largest value.

Input Format: First line contains integer t which is number of test case. <br>

For each test case, you are given a single integer n in the first line which is the size of array A[] and next line contains n space separated integers denoting the elements of the array A .

Sample Input: 1

4

54 546 548 60

Output Format: Print the largest value.

Sample Output: 6054854654

```
// Given an array of numbers, program to arrange the numbers to form the
```

```
// largest number
```

```
#include <iostream>
```

```
#include <string>
```

```
#include <vector>
```

```
#include <algorithm>
```

```
using namespace std;
```

```
// A comparison function which is used by sort() in printLargest()
```

```
int myCompare(string X, string Y)
```

```
{
```

```
 // first append Y at the end of X
```

```
 string XY = X.append(Y);
```

```
 // then append X at the end of Y
```

```
 string YX = Y.append(X);
```

```
 // Now see which of the two formed numbers is greater
```

```
 return XY.compare(YX) > 0 ? 1: 0;
```

```
}
```

```
// The main function that prints the arrangement with the largest value.
```

```
// The function accepts a vector of strings
```

```

void printLargest(vector<string> arr)
{
 // Sort the numbers using library sort funtion. The function uses
 // our comparison function myCompare() to compare two strings.
 // See http://www.cplusplus.com/reference/algorithm/sort/ for details
 sort(arr.begin(), arr.end(), myCompare);

 for (int i=0; i < arr.size() ; i++)
 cout << arr[i];
 cout<<"\n";
}

// driverr program to test above functions
int main()
{
 int t;
 cin>>t;
 for(int i=0;i<t;i++)
 {
 vector<string> arr;
 int n;
 cin>>n;
 for(int j=0;j<n;j++)
 {
 string s;
 cin>>s;
 arr.push_back(s);
 }
 }
}

```

```

 printLargest(arr);
 }

 // vector<string> arr;

 // //output should be 6054854654
 // arr.push_back("54");
 // arr.push_back("546");
 // arr.push_back("548");
 // arr.push_back("60");
 // printLargest(arr);

 // output should be 777776
 /*arr.push_back("7");
 arr.push_back("776");
 arr.push_back("7");
 arr.push_back("7");*/

 //output should be 998764543431
 /*arr.push_back("1");
 arr.push_back("34");
 arr.push_back("3");
 arr.push_back("98");
 arr.push_back("9");
 arr.push_back("76");
 arr.push_back("45");
 arr.push_back("4");
 */

 return 0;

```

```
}
```

Problem Name: Chewbacca and Number

Problem Difficulty: 1

Problem Constraints:  $x \leq 1000000000000000000$

Problem Description:

Luke Skywalker gave Chewbacca an integer number  $x$ . Chewbacca isn't good at numbers but he loves inverting digits in them. Inverting digit  $t$  means replacing it with digit  $9 - t$ .

Help Chewbacca to transform the initial number  $x$  to the minimum possible positive number by inverting some (possibly, zero) digits. The decimal representation of the final number shouldn't start with a zero.

Input Format: The first line contains a single integer  $x$  ( $1 \leq x \leq 10^{18}$ ) — the number that Luke Skywalker gave to Chewbacca.

Sample Input: 4545

Output Format: Print the minimum possible positive number that Chewbacca can obtain after inverting some digits. The number shouldn't contain leading zeroes.

Sample Output: 4444

```
#include<iostream>

using namespace std;
```

```
int main()
{
 char a[50];

 cin>>a;

 int i=0;
```



```

if(a[i]=='9')
{
 i++;
}

//iterate over the remaining characte till not found null character
for(;a[i]!='\0';i++)
{
 int digit=a[i]-'0'; //convert char into a int
 if(digit>=5)
 {
 digit=9-digit;
 a[i]=digit + '0'; //again int to char
 }

}

cout<<a<<endl;

return 0;
}

```

PMO gives two random numbers a & b to the Prime Minister. PM Modi wants to visit all countries between a and b (inclusive) , However due to shortage of time he can't visit each and every country , So PM Modi decides to visit only those countries,for which country number has two divisors. So your task is to find number of countries Mr.Modi will visit.

### **Input Format**

The first line contains N , no of test cases. The next lines contain two integers a and b denoting the range of country numbers.

### **Constraints**

$a \leq 1000000$  &  $b \leq 1000000$ .  $N \leq 1000$

### **Output Format**

Output contains N lines each containing an answer for the test case.

### **Sample Input**

2

1 10

11 20

### **Sample Output**

4

4

### **Explanation**

Problem Name: Broken Calculator

Problem Difficulty: None

Problem Constraints:  $1 \leq N \leq 500$

Problem Description:

Andrew was attempting a mathematics test where he needed to solve problems with factorials.

One such problem had an answer which equaled  $100!$  ,He wondered what would this number look like.

He tried to calculate 100! On his scientific calculator but failed to get a correct answer. Can you write a code to help Andrew calculate factorials of such large numbers?

Input Format: a single lined integer N

Sample Input: 5

Output Format: Print the factorial of N

Sample Output: 120

=====Solution=====

```
#include<iostream>
```

```
using namespace std;
```

```
#define MAX 2000
```

```
int multiply(int x, int res[], int res_size);
```

```
void factorial(int n)
```

```
{
```

```
 int res[MAX];
```

```
 res[0] = 1;
```

```
 int res_size = 1;
```

```
 for (int x=2; x<=n; x++)
```

```
 {
```

```

 res_size = multiply(x, res, res_size);
 }

 //cout << "Factorial of given number is \n";
 for (int i=res_size-1; i>=0; i--)
 cout << res[i];
}

```

```

int multiply(int x, int res[], int res_size)
{
 int carry = 0;

 for (int i=0; i<res_size; i++)
 {
 int prod = res[i] * x + carry;
 res[i] = prod % 10;
 carry = prod/10;
 }
}

```

```

while (carry)
{
 res[res_size] = carry%10;
 carry = carry/10;
 res_size++;
}

```

```
 return res_size;
}
```

```
int main()
{
 int n;
 cin>>n;
 factorial(n);
 return 0;
}
```

Problem Name: Matrix Search

Problem Difficulty: None

Problem Constraints:  $1 \leq N, M \leq 30$

$0 \leq A[i] \leq 100$

Problem Description:

Given an  $n \times m$  matrix, where every row and column is sorted in increasing order, and a number  $x$ . Find if element  $x$  is present in the matrix or not.

Input Format: First line consists of two space separated integers  $N$  and  $M$ , denoting the number of element in a row and column respectively. Second line of each test case consists of  $N \times M$  space separated integers denoting the elements in the matrix in row major order. Third line of each test case contains a single integer  $x$ , the element to be searched.

Sample Input: 3 3

3 30 38

44 52 54

57 60 69

62

Output Format: Print 1 if the element is present in the matrix, else 0.

Sample Output: 0

=====Solution=====

```
#include <iostream>
```

```
using namespace std;
```

```
void matrixSearch(int **arr, int n , int m, int key) {
 for(int row = 0;row < n;) {
 for(int col=m-1;col>=0 and row < n;) {
 if(arr[row][col] == key) {
 cout<<1;
 return;
 } else if(arr[row][col] > key) {
 col--;
 } else {
 row++;
 }
 }
 }
 cout<<0;
 return ;
}
```

```
int main() {
 int n, m;
 cin>>n>>m;
 int **arr = new int*[n];
```

```
for(int i=0;i<n;i++) {
 arr[i]=new int[m];
}
for(int i=0;i<n;i++) {
 for(int j=0;j<m;j++) {
 cin>>arr[i][j];
 }
}
int key;
cin>>key;
matrixSearch(arr, n, m, key);
}
```

Problem Name: Arrays-Sum Of Two Arrays

Problem Difficulty: None

Problem Constraints: Length of Array should be between 1 and 1000.

Problem Description:

Take as input N, the size of array. Take N more inputs and store that in an array. Take as input M, the size of second array and take M more inputs and store that in second array. Write a function that returns the sum of two arrays. Print the value returned.

Input Format:

Sample Input: 4

1 0 2 9

5

3 4 5 6 7

Output Format:

Sample Output: 3, 5, 5, 9, 6, END

=====Solution=====

```
#include <iostream>
```

```
using namespace std;
```

```
void print(int num)
```

```
{
```

```
 if(num == 0)
```

```
 {
```

```
 return;
```

```
 }
```

```
 print(num / 10);
```

```
 cout << num % 10;
```

```
 cout << ", ";
```

```
}
```

```
int main()
```

```
{
```

```
 int num1[1000] , num2[1000];
```



```
int n;
```

```
cin >> n;
```

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

```
{
```

```
 cin >> num1[i];
```

```
}
```

```
int m;
```

```
cin >> m;
```

```
for(int j = 0 ; j < m ; j++)
```

```
{
```

```
 cin >> num2[j];
```

```
}
```

```
int fnum = 0;
```

```
int snum = 0;
```

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

```
{
```

```
 fnum = fnum * 10 + num1[i];
```

```
}
```

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

```
{
```

```
 snum = snum * 10 + num2[i];
```

```
}
```

```
int sum = fnum + snum;

print(sum);

cout << "END";
}
```

Problem Name: Median of Sorted Arrays

Problem Difficulty: None

Problem Constraints:  $N < 1000$

Problem Description:

We are given two sorted arrays of same size  $n$ . Find the median of an array formed after merging these two sorted arrays.

Input Format: First line contains the input  $n$ . Second and Third line contains  $n$  space separated integers.

Sample Input: 5

1 3 5 7 9

2 4 6 8 10

Output Format: Print the median in a single line.

Sample Output: 5

=====Solution=====

```
#include<iostream>
```

```
using namespace std;
```

```
int median(int *arr1,int *arr2,int n){
```

```
int lo1 = 0,hi1 = n-1;
```

```
int lo2 = 0,hi2 = n-1;
```

```
if(n == 1){
```

```
 return (arr1[0] + arr2[0]) >> 1;
```

```
}
```

```
while(true){
```

```
 int m1,m2,median1,median2;
```

```
 median1 = arr1[(hi1 + lo1)>>1];
```

```
 median2 = arr2[(hi2 + lo2)>>1];
```

```
 m1 = (hi1 + lo1)>>1;
```

```
 m2 = (hi2 + lo2)>>1;
```

```
 if(hi1 - lo1 == 1){
```

```
 return (max(arr1[lo1] , arr2[lo1]) +
```

```
 min(arr1[hi1] , arr2[hi2])) / 2;
```

```
}
```

```
if(median1 == median2)
```

```
 return median1;
```

```
if(m1 > m2){
```

```
 hi1 = m1;
 lo2 = m2;
 }
 else{
 hi2 = m2;
 lo1 = m1;

 }
}
return 0;
}

int main(){
 int arr1[1000],arr2[1000];
 int n ;
 cin>>n;
 for(int i=0;i<n;i++){
 cin>>arr1[i];
 }

 for(int i=0;i<n;i++){
 cin>>arr2[i];
 }

 cout<<median(arr1,arr2,n)<<endl;
 return 0;
```