# Array

## [Subarray with given sum](https://practice.geeksforgeeks.org/problems/subarray-with-given-sum/0)

Given an unsorted array of nonnegative integers, find a continuous subarray which adds to a given number.

**Examples :**

***Input****: arr[] = {1, 4, 20, 3, 10, 5}, sum = 33****Ouptut****: Sum found between indexes 2 and 4*

***Input****: arr[] = {1, 4, 0, 0, 3, 10, 5}, sum = 7****Ouptut****: Sum found between indexes 1 and 4*

***Input****: arr[] = {1, 4}, sum = 0****Output****: No subarray found*

#### Approach: Initialize a variable curr\_sum as the first element. curr\_sum indicates the sum of the current subarray. Start from the second element and add all elements one by one to the curr\_sum. If curr\_sum becomes equal to the sum, then print the solution. If curr\_sum exceeds the sum, then remove trailing elements while curr\_sum is greater than the sum.

/\* An efficient program to print

subarray with sum as given sum \*/

#include <iostream>

using namespace std;

/\* Returns true if the there is a subarray of

arr[] with a sum equal to 'sum' otherwise

returns false. Also, prints the result \*/

int subArraySum(int arr[], int n, int sum)

{

    /\* Initialize curr\_sum as value of

    first element and starting point as 0 \*/

    int curr\_sum = arr[0], start = 0, i;

    /\* Add elements one by one to curr\_sum and

    if the curr\_sum exceeds the sum,

    then remove starting element \*/

    for (i = 1; i <= n; i++)

    {

        // If curr\_sum exceeds the sum,

        // then remove the starting elements

        while (curr\_sum > sum && start < i - 1)

        {

            curr\_sum = curr\_sum - arr[start];

            start++;

        }

        // If curr\_sum becomes equal to sum,

        // then return true

        if (curr\_sum == sum)

        {

            cout << "Sum found between indexes "

                 << start << " and " << i - 1;

            return 1;

        }

        // Add this element to curr\_sum

        if (i < n)

        curr\_sum = curr\_sum + arr[i];

    }

    // If we reach here, then no subarray

    cout << "No subarray found";

    return 0;

}

Approach 2:

#include<bits/stdc++.h>

using namespace std;

// Function to print subarray with sum as given sum

void subArraySum(int arr[], int n, int sum)

{

    // create an empty map

    unordered\_map<int, int> map;

    // Maintains sum of elements so far

    int curr\_sum = 0;

    for (int i = 0; i < n; i++)

    {

        // add current element to curr\_sum

        curr\_sum = curr\_sum + arr[i];

        // if curr\_sum is equal to target sum

        // we found a subarray starting from index 0

        // and ending at index i

        if (curr\_sum == sum)

        {

            cout << "Sum found between indexes "

                 << 0 << " to " << i << endl;

            return;

        }

        // If curr\_sum - sum already exists in map

        // we have found a subarray with target sum

        if (map.find(curr\_sum - sum) != map.end())

        {

            cout << "Sum found between indexes "

                 << map[curr\_sum - sum] + 1

                 << " to " << i << endl;

            return;

        }

        map[curr\_sum] = i;

    }

    // If we reach here, then no subarray exists

    cout << "No subarray with given sum exists";

}

## [Count the triplets](https://practice.geeksforgeeks.org/problems/count-the-triplets/0)

**Difficulty:**[**Easy**](https://practice.geeksforgeeks.org/Easy/0/0/)**Marks: 2**

#### Approach: Sort array, pick j=0 and k=i-1 where i=n-1 to 1 if triplet found, increase counter

Show Topic Tags

Company Tags [Amazon](https://practice.geeksforgeeks.org/company/Amazon/)  [Arcesium](https://practice.geeksforgeeks.org/company/Arcesium/)

* [**Problems**](https://practice.geeksforgeeks.org/problems/count-the-triplets/0#problems)

Given an array of distinct integers. The task is to count all the triplets such that sum of two elements equals the third element.

**Input:**  
The first line of input contains an integer **T** denoting the number of test cases. Then T test cases follow. Each test case consists of two lines. First line of each test case contains an Integer **N** denoting size of array and the second line contains N space separated elements.

**Output:**  
For each test case, print the count of all triplets, in new line. If no such triplets can form, print "**-1**".

**Constraints:**  
1 <= T <= 100  
3 <= N <= 105  
1 <= A[i] <= 106

**Example:  
Input:**

2  
4  
1 5 3 2  
3  
3 2 7  
**Output:**  
2  
-1

**Explanation:  
Testcase 1:** There are 2 triplets: 1 + 2 = 3 and 3 +2 = 5

#include <iostream>

#include <vector>

#include<bits/stdc++.h>

using namespace std;

int countTriplets(vector<int> &A, int N){

if(N<3) return -1;

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

int count=0;

for(int i=N-1; i>1; i--){

int j=0, k=i-1;

while(j<k){

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

count++;

j++;

k--;

}

else if(A[j]+A[k]<A[i])

j++;

else

k--;

}

}

if(count>0)

return count;

return count=-1;

}

int main() {

//code

int t, N;

vector<int> A;

cin>>t;

while(t--){

cin>>N;

A.resize(N);

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

cin>>A[i];

cout<<countTriplets(A,N)<<endl;

A.clear();

}

return 0;

}

## [Kadane’s Algorithm](https://practice.geeksforgeeks.org/problems/kadanes-algorithm/0)

Company Tags [24\*7 Innovation Labs](https://practice.geeksforgeeks.org/company/24*7%20Innovation%20Labs/)  [Accolite](https://practice.geeksforgeeks.org/company/Accolite/)  [Amazon](https://practice.geeksforgeeks.org/company/Amazon/)  [Citrix](https://practice.geeksforgeeks.org/company/Citrix/)  [D-E-Shaw](https://practice.geeksforgeeks.org/company/D-E-Shaw/)  [FactSet](https://practice.geeksforgeeks.org/company/FactSet/)  [Flipkart](https://practice.geeksforgeeks.org/company/Flipkart/)  [Hike](https://practice.geeksforgeeks.org/company/Hike/)  [Housing.com](https://practice.geeksforgeeks.org/company/Housing.com/)  [MakeMyTrip](https://practice.geeksforgeeks.org/company/MakeMyTrip/)  [MetLife](https://practice.geeksforgeeks.org/company/MetLife/)  [Microsoft](https://practice.geeksforgeeks.org/company/Microsoft/)  [Morgan Stanley](https://practice.geeksforgeeks.org/company/Morgan%20Stanley/)  [Ola Cabs](https://practice.geeksforgeeks.org/company/Ola%20Cabs/)  [Oracle](https://practice.geeksforgeeks.org/company/Oracle/)  [OYO Rooms](https://practice.geeksforgeeks.org/company/OYO%20Rooms/)  [Payu](https://practice.geeksforgeeks.org/company/Payu/)  [Samsung](https://practice.geeksforgeeks.org/company/Samsung/)  [Snapdeal](https://practice.geeksforgeeks.org/company/Snapdeal/)  [Teradata](https://practice.geeksforgeeks.org/company/Teradata/)  [Visa](https://practice.geeksforgeeks.org/company/Visa/)  [VMWare](https://practice.geeksforgeeks.org/company/VMWare/)  [Walmart](https://practice.geeksforgeeks.org/company/Walmart/)  [Zoho](https://practice.geeksforgeeks.org/company/Zoho/)

[**Problems**](https://practice.geeksforgeeks.org/problems/kadanes-algorithm/0#problems)

Given an array **arr**of **N** integers. Find the contiguous sub-array with maximum sum.

**Input:**  
The first line of input contains an integer **T**denoting the number of test cases. The description of **T**test cases follows. The first line of each test case contains a single integer **N**denoting the size of array. The second line contains **N**space-separated integers **A1, A2, ..., AN** denoting the elements of the array**.**

**Output:**  
Print the maximum sum of the contiguous sub-array in a separate line for each test case.

**Constraints:**  
1 ≤ T ≤ 110  
1 ≤ N ≤ 106  
-107 ≤ A[i] <= 107

**Example:**  
**Input**  
2  
5  
1 2 3 -2 5  
4  
-1 -2 -3 -4  
**Output**  
9  
-1

**Explanation:  
Testcase 1:** Max subarray sum is 9 of elements (1, 2, 3, -2, 5) which is a contiguous subarray.

#include <bits/stdc++.h>

using namespace std;

int maxSum(int \*a, int n){

int csum, msum;

csum=msum=a[0];

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

if(csum+a[i]>a[i])

csum=csum+a[i];

else csum=a[i];

msum = max(msum, csum);

}

return msum;

}

int main() {

//code

int t, n, \*a;

cin>>t;

while(t--){

cin>>n;

a = new int[n];

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

cin>>a[i];

cout<<maxSum(a, n)<<endl;

}

return 0;

}

## [Missing number in array](https://practice.geeksforgeeks.org/problems/missing-number-in-array/0)

Company Tags [Accolite](https://practice.geeksforgeeks.org/company/Accolite/)  [Adobe](https://practice.geeksforgeeks.org/company/Adobe/)  [Amazon](https://practice.geeksforgeeks.org/company/Amazon/)  [Cisco](https://practice.geeksforgeeks.org/company/Cisco/)  [D-E-Shaw](https://practice.geeksforgeeks.org/company/D-E-Shaw/)  [Intuit](https://practice.geeksforgeeks.org/company/Intuit%20/)  [Microsoft](https://practice.geeksforgeeks.org/company/Microsoft/)  [Morgan Stanley](https://practice.geeksforgeeks.org/company/Morgan%20Stanley/)  [Ola Cabs](https://practice.geeksforgeeks.org/company/Ola%20Cabs/)  [Payu](https://practice.geeksforgeeks.org/company/Payu/)  [Qualcomm](https://practice.geeksforgeeks.org/company/Qualcomm/)  [Visa](https://practice.geeksforgeeks.org/company/Visa/)

[**Problems**](https://practice.geeksforgeeks.org/problems/missing-number-in-array/0#problems)

Given an array **C** of size **N-1** and given that there are numbers from **1** to **N** with one element missing, the missing number is to be found.

**Input:**  
The first line of input contains an integer **T** denoting the number of test cases. For each test case first line contains **N**(size of array). The subsequent line contains N-1 array elements.

**Output:**  
Print the missing number in array.

**Constraints:**  
1 ≤ T ≤ 200  
1 ≤ N ≤ 107  
1 ≤ C[i] ≤ 107

**Example:**  
**Input:**  
2  
5  
1 2 3 5  
10  
1 2 3 4 5 6 7 8 10

**Output:**  
4  
9

**Explanation:**  
**Testcase 1:** Given array : 1 2 3 5. Missing element is 4.

#include <iostream>

#include <vector>

#include <bits/stdc++.h>

using namespace std;

int findMissing(vector<int> &A, int n){

if(n<1) return -1;

int sum1=n\*(n+1)/2;

int sum2= accumulate(A.begin(), A.end(), 0);

return (sum1-sum2);

}

int main() {

//code

int t,n;

vector<int> A;

cin>>t;

while(t--){

cin>>n;

A.resize(n-1);

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

cin>>A[i];

cout<<findMissing(A,n)<<endl;

}

return 0;

}

## [Merge two sorted arrays](https://practice.geeksforgeeks.org/problems/merge-two-sorted-arrays/0/)

#include <iostream>

#include <algorithm>

using namespace std;

// Utility function to print contents of an array

void printArray(int arr[], int n){

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

cout << arr[i] << " ";

}

cout << '\n';

}

// in-place merge two sorted arrays X[] and Y[]

// invariant: X[] and Y[] are sorted at any point

void merge(int X[], int Y[], int m, int n){

// consider each element X[i] of array X and ignore the element

// if it is already in correct order else swap it with next smaller

// element which happens to be first element of Y

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

// compare current element of X[] with first element of Y[]

if (X[i] > Y[0]) {

swap(X[i], Y[0]);

int first = Y[0];

// move Y[0] to its correct position to maintain sorted

// order of Y[]. Note: Y[1..n-1] is already sorted

int k;

for (k = 1; k < n && Y[k] < first; k++) {

Y[k - 1] = Y[k];

}

Y[k - 1] = first;

}

}

}

void merge(int \*arr1, int \*arr2, int n, int m)

{

    int i, j, gap = n + m;

    for (gap = nextGap(gap); gap > 0; gap = nextGap(gap))

    {

        // comparing elements in the first array.

        for (i = 0; i + gap < n; i++)

            if (arr1[i] > arr1[i + gap])

                swap(arr1[i], arr1[i + gap]);

        //comparing elements in both arrays.

        for (j = gap > n ? gap-n : 0 ; i < n&&j < m; i++, j++)

            if (arr1[i] > arr2[j])

                swap(arr1[i], arr2[j]);

        if (j < m)

        {

            //comparing elements in the second array.

            for (j = 0; j + gap < m; j++)

                if (arr2[j] > arr2[j + gap])

                    swap(arr2[j], arr2[j + gap]);

        }

    }

}

// main function

## [Rearrange array alternatively](https://practice.geeksforgeeks.org/problems/-rearrange-array-alternately/0/)

## [Number of pairs](https://practice.geeksforgeeks.org/problems/number-of-pairs/0/)

## [Inversion of Array](https://practice.geeksforgeeks.org/problems/inversion-of-array/0/)

## Count Inversions in an array | Set 1 (Using Merge Sort)

Inversion Count for an array indicates – how far (or close) the array is from being sorted. If array is already sorted then inversion count is 0. If array is sorted in reverse order that inversion count is the maximum.  
Formally speaking, two elements a[i] and a[j] form an inversion if a[i] > a[j] and i < j

**Example:**  
The sequence 2, 4, 1, 3, 5 has three inversions (2, 1), (4, 1), (4, 3).

**const** **int** mod **=** 1000000007;

**long** merge(vector**<int>&**left, vector**<int>&** right, vector**<int>&**A){

**int** l**=**0,r**=**0,lsize**=**left.size(),rsize**=**right.size();

**long** smaller**=**0,count**=**0;

**while**(l**<**lsize **&&** r**<**rsize){

**if**(left[l]**<=**right[r]){

A.push\_back(left[l**++**]);

count **=** (count**%**mod) **+** (smaller**%**mod);

count **%=** mod;

}

**else**{

A.push\_back(right[r**++**]);

smaller**++**;

smaller **%=** mod;

}

}

**while**(l**<**lsize){

A.push\_back(left[l**++**]);

count **=** (count**%**mod) **+** (smaller**%**mod);

count **%=** mod;

}

**while**(r**<**rsize){

A.push\_back(right[r**++**]);

}

**return** count;

}

**long** merge\_sort(vector**<int>&** A){

**int** size **=** A.size();

**if**(size**<=**1) **return** 0;

vector**<int>** left, right;

left.assign(A.begin(), A.begin()**+**size**/**2);

right.assign(A.begin()**+**size**/**2, A.end());

**long** count **=** 0;

A.clear();

count **=** (count**%**mod) **+** (merge\_sort(left)**%**mod);

count **%=** mod;

count **=** (count**%**mod) **+** (merge\_sort(right)**%**mod);

count **%=** mod;

count **=** (count**%**mod) **+** (merge(left, right, A)**%**mod);

count **%=** mod;

**return** count;

}

**int** Solution**::**solve(vector**<int>** **&**A){

*/// naive way is O(N^2)*

*/// we can further optimize it to O(NlogN)*

**return** merge\_sort(A);

}

## [Sort an array of 0s, 1s and 2s](https://practice.geeksforgeeks.org/problems/sort-an-array-of-0s-1s-and-2s/0)

**Sort an array of 0s, 1s and 2s**

Company Tags [Adobe](https://practice.geeksforgeeks.org/company/Adobe/)  [Amazon](https://practice.geeksforgeeks.org/company/Amazon/)  [Hike](https://practice.geeksforgeeks.org/company/Hike/)  [MakeMyTrip](https://practice.geeksforgeeks.org/company/MakeMyTrip/)  [MAQ Software](https://practice.geeksforgeeks.org/company/MAQ%20Software/)  [Microsoft](https://practice.geeksforgeeks.org/company/Microsoft/)  [Morgan Stanley](https://practice.geeksforgeeks.org/company/Morgan%20Stanley/)  [Ola Cabs](https://practice.geeksforgeeks.org/company/Ola%20Cabs/)  [OYO Rooms](https://practice.geeksforgeeks.org/company/OYO%20Rooms/)  [Paytm](https://practice.geeksforgeeks.org/company/Paytm/)  [Qualcomm](https://practice.geeksforgeeks.org/company/Qualcomm/)  [Samsung](https://practice.geeksforgeeks.org/company/Samsung/)  [SAP Labs](https://practice.geeksforgeeks.org/company/SAP%20Labs/)  [Snapdeal](https://practice.geeksforgeeks.org/company/Snapdeal/)  [Walmart](https://practice.geeksforgeeks.org/company/Walmart/)  [Yatra.com](https://practice.geeksforgeeks.org/company/Yatra.com/)

* [**Problems**](https://practice.geeksforgeeks.org/problems/sort-an-array-of-0s-1s-and-2s/0#problems)

Given an array **A**of size **N** containing**0s, 1s, and 2s**; you need to sort the array in ascending order.

**Input:**  
The first line contains an integer **'T'** denoting the total number of test cases. Then **T**testcases follow. Each testcases contains two lines of input. The first line denotes the size of the array **N**. The second lines contains the elements of the array A separated by spaces.  
  
**Output:**  
For each testcase, print the sorted array.  
  
**Constraints:**  
1 <= T <= 500  
1 <= N <= 106  
0 <= Ai <= 2

**Example:  
Input :**  
2  
5  
0 2 1 2 0  
3  
0 1 0

**Output:**  
0 0 1 2 2  
0 0 1

**Explanation:**  
**Testcase 1:** After segregating the 0s, 1s and 2s, we have 0 0 1 2 2 which shown in the output.

using namespace std;

void swap(int \*a, int \*b){

int t = \*a;

\*a = \*b;

\*b = t;

}

void sortA(int \*a, int n){

int low = 0, high = n -1, mid = 1;

while(mid <= high){

switch(a[mid]){

case 0:

if(mid == low){

low++;

mid++;

}

else{

swap(&a[low], &a[mid]);

low++;

}

break;

case 1:

mid++;

break;

case 2:

swap(&a[mid], &a[high]);

high--;

break;

}

}

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

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

cout<<endl;

}

int main() {

//code

int T, N, \*a;

cin>>T;

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

cin>>N;

a = new int[N];

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

cin>>a[j];

sortA(a, N);

}

return 0;

}

## [Equilibrium point](https://practice.geeksforgeeks.org/problems/equilibrium-point/0)

**Equilibrium point**

Company Tags [Accolite](https://practice.geeksforgeeks.org/company/Accolite/)  [Adobe](https://practice.geeksforgeeks.org/company/Adobe/)  [Amazon](https://practice.geeksforgeeks.org/company/Amazon/)  [Hike](https://practice.geeksforgeeks.org/company/Hike/)  [Zoho](https://practice.geeksforgeeks.org/company/Zoho/)

* [**Problems**](https://practice.geeksforgeeks.org/problems/equilibrium-point/0#problems)

Given an array **A** of **N** positive numbers. The task is to find the position where equilibrium first occurs in the array. Equilibrium position in an array is a position such that the sum of elements before it is equal to the sum of elements after it.

**Input:**  
The first line of input contains an integer **T**, denoting the number of test cases. Then T test cases follow. First line of each test case contains an integer N denoting the size of the array. Then in the next line are N space separated values of the array A.

**Output:**  
For each test case in a new  line print the position at which the elements are at equilibrium if no equilibrium point exists print -1.

**Constraints:**  
1 <= T <= 100  
1 <= N <= 106  
1 <= Ai <= 108

**Example:  
Input:**  
2  
1  
1  
5  
1 3 5 2 2

**Output:**  
1  
3

**Explanation:**  
**Testcase 1:** Since its the only element hence its the only equilibrium point.  
**Testcase 2:** For second test case equilibrium point is at position 3 as elements below it (1+3) = elements after it (2+2).

#include <iostream>

#include<bits/stdc++.h>

using namespace std;

int equillibrium(vector<int> &A, int N){

if(N<1) return -1;

int sumR=accumulate(A.begin(), A.end(), 0);

int sumL=0;

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

sumR-=A[i];

if(sumL==sumR)

return i+1;

sumL+=A[i];

}

return -1;

}

int main() {

//code

int t,N;

vector<int>A;

cin>>t;

while(t--){

cin>>N;

A.resize(N);

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

cin>>A[i];

cout<<equillibrium(A,N)<<endl;

}

return 0;

}

## [Leaders in an array](https://practice.geeksforgeeks.org/problems/leaders-in-an-array/0)

**Leaders in an array**

Company Tags [Amazon](https://practice.geeksforgeeks.org/company/Amazon/)  [Payu](https://practice.geeksforgeeks.org/company/Payu/)  [Samsung](https://practice.geeksforgeeks.org/company/Samsung/)

* [**Problems**](https://practice.geeksforgeeks.org/problems/leaders-in-an-array/0#problems)

Given an array of positive integers. Your task is to find the leaders in the array.  
**Note:** An element of array is leader if it is greater than or equal to all the elements to its right side. Also, the rightmost element is always a leader.

**Input:**  
The first line of input contains an integer **T** denoting the number of test cases. The description of **T** test cases follows.  
The first line of each test case contains a single integer **N** denoting the size of array.  
The second line contains N space-separated integers A1, A2, ..., AN denoting the elements of the array.

**Output:**  
Print all the leaders.

**Constraints:**  
1 <= T <= 100  
1 <= N <= 107  
0 <= Ai <= 107

**Example:**  
**Input:**  
3  
6  
16 17 4 3 5 2  
5  
1 2 3 4 0  
5  
7 4 5 7 3  
**Output:**  
17 5 2  
4 0  
7 7 3

**Explanation:**  
**Testcase 3:** All elements on the right of 7 (at index 0) are smaller than or equal to 7. Also, all the elements of right side of 7 (at index 3) are smaller than 7. And, the last element 3 is itself a leader since no elements are on its right.

#include <iostream>

#include <bits/stdc++.h>

using namespace std;

void print (vector<int> &A){

for(auto it=A.begin(); it!=A.end(); ++it)

cout<<\*it<<" ";

cout<<endl;

}

void leaders(vector<int> &A, int n){

if(n<1) return;

vector<int> res;

int leader=A[n-1];

res.push\_back(leader);

//cout<<leader<<" ";

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

if(A[i]>=leader){

leader=A[i];

res.push\_back(leader);

//cout<<leader<<" ";

}

}

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

print(res);

}

int main() {

//code

int t,n;

vector<int> A;

cin>>t;

while(t--){

cin>>n;

A.resize(n);

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

cin>>A[i];

leaders(A,n);

A.clear();

}

return 0;

}

## [Minimum Platforms](https://practice.geeksforgeeks.org/problems/minimum-platforms/0)

**Minimum Platforms**

Company Tags [Accolite](https://practice.geeksforgeeks.org/company/Accolite/)  [Amazon](https://practice.geeksforgeeks.org/company/Amazon/)  [Boomerang Commerce](https://practice.geeksforgeeks.org/company/Boomerang%20Commerce/)  [D-E-Shaw](https://practice.geeksforgeeks.org/company/D-E-Shaw/)  [Directi](https://practice.geeksforgeeks.org/company/Directi/)  [Hike](https://practice.geeksforgeeks.org/company/Hike/)  [Morgan Stanley](https://practice.geeksforgeeks.org/company/Morgan%20Stanley/)  [OYO Rooms](https://practice.geeksforgeeks.org/company/OYO%20Rooms/)  [Paytm](https://practice.geeksforgeeks.org/company/Paytm/)  [Walmart](https://practice.geeksforgeeks.org/company/Walmart/)  [Zillious](https://practice.geeksforgeeks.org/company/Zillious/)

* [**Problems**](https://practice.geeksforgeeks.org/problems/minimum-platforms/0#problems)

Given arrival and departure times of all trains that reach a railway station. Your task is to find the minimum number of platforms required for the railway station so that no train waits.

**Note:** Consider that all the trains arrive on the same day and leave on the same day. Also, arrival and departure times will not be same for a train, but we can have arrival time of one train equal to departure of the other.

In such cases, **we need different platforms,**i.e at any given instance of time, **same platform can not be used for both departure of a train and arrival of another.**

**Input:**  
The first line of input contains T, the number of test cases. For each test case, first line will contain an integer N, the number of trains. Next two lines will consist of **N** space separated time intervals denoting arrival and departure times respectively.  
**Note:** Time intervals are in the 24-hour format(hhmm),  of the for **HHMM ,** where the first two charcters represent hour (between 00 to 23 ) and last two characters represent minutes (between 00 to 59).

**Output:**  
For each test case, print the minimum number of platforms required for the trains to arrive and depart safely.

**Constraints:**  
1 <= T <= 100  
1 <= N <= 1000  
1 <= A[i] < D[i] <= 2359

**Example:**  
**Input:**  
2  
6   
0900  0940 0950  1100 1500 1800  
0910 1200 1120 1130 1900 2000  
3  
0900 1100 1235  
1000 1200 1240

**Output:**  
3  
1

**Explanation:**  
**Testcase 1:**Minimum 3 platforms are required to safely arrive and depart all trains.

// Program to find minimum number of platforms

// required on a railway station

#include <bits/stdc++.h>

using namespace std;

int findPlatform(int arr[], int dep[], int n)

{

    // Insert all the times (arr. and dep.)

    // in the multimap.

    multimap<int, char> order;

    for (int i = 0; i < n; i++) {

        // If its arrival then second value

        // of pair is 'a' else 'd'

        order.insert(make\_pair(arr[i], 'a'));

        order.insert(make\_pair(dep[i], 'd'));

    }

    int result = 0;

    int plat\_needed = 0;

    multimap<int, char>::iterator it = order.begin();

    // Start iterating the multimap.

    for (; it != order.end(); it++) {

        // If its 'a' then add 1 to plat\_needed

        // else minus 1 from plat\_needed.

        if ((\*it).second == 'a')

            plat\_needed++;

        else

            plat\_needed--;

        if (plat\_needed>result)

            result = plat\_needed;

    }

    return result;

}

// Driver code

int main()

{

    int arr[] = { 900, 940, 950, 1100, 1500, 1800 };

    int dep[] = { 910, 1200, 1120, 1130, 1900, 2000 };

    int n = sizeof(arr) / sizeof(arr[0]);

    cout << "Minimum Number of Platforms Required = "

         << findPlatform(arr, dep, n);

    return 0;

}

## [Reverse array in groups](https://practice.geeksforgeeks.org/problems/reverse-array-in-groups/0)

**Reverse array in groups**

Company Tags [Amazon](https://practice.geeksforgeeks.org/company/Amazon/)

* [**Problems**](https://practice.geeksforgeeks.org/problems/reverse-array-in-groups/0#problems)

Given an array **arr[]** of positive integers of size **N**. Reverse every sub-array of **K**group elements.

**Input:**  
The first line of input contains a single integer **T** denoting the number of test cases. Then**T** test cases follow. Each test case consist of two lines of input. The first line of each test case consists of an integer **N**(size of array) and an integer **K** separated by a space. The second line of each test case contains**N** space separated integers denoting the array elements.

**Output:**  
For each test case, print the modified array.

**Constraints:**  
1 ≤ T ≤ 200  
1 ≤ N, K ≤ 107  
1 ≤ A[i] ≤ 1018

**Example:**  
**Input**  
2  
5 3  
1 2 3 4 5  
6 2  
10 20 30 40 50 60

**Output**  
3 2 1 5 4  
20 10 40 30 60 50

**Explanation:  
Testcase 1:** Reversing groups in size 3, first group consists of elements 1, 2, 3. Reversing this group, we have elements in order as 3, 2, 1.

// Function to reverse every sub-array formed by

// consecutive k elements

void reverse(int arr[], int n, int k)

{

    for (int i = 0; i < n; i += k)

    {

        int left = i;

        // to handle case when k is not multiple of n

        int right = min(i + k - 1, n - 1);

        // reverse the sub-array [left, right]

        while (left < right)

            swap(arr[left++], arr[right--]);

    }

}

## [K’th smallest element](https://practice.geeksforgeeks.org/problems/kth-smallest-element/0)

**Kth smallest element**

Company Tags [ABCO](https://practice.geeksforgeeks.org/company/ABCO/)  [Accolite](https://practice.geeksforgeeks.org/company/Accolite/)  [Amazon](https://practice.geeksforgeeks.org/company/Amazon/)  [Cisco](https://practice.geeksforgeeks.org/company/Cisco/)  [Hike](https://practice.geeksforgeeks.org/company/Hike/)  [Microsoft](https://practice.geeksforgeeks.org/company/Microsoft/)  [Snapdeal](https://practice.geeksforgeeks.org/company/Snapdeal/)  [VMWare](https://practice.geeksforgeeks.org/company/VMWare/)

* [**Problems**](https://practice.geeksforgeeks.org/problems/kth-smallest-element/0#problems)

Given an array **arr[]** and a number **K** where K is smaller than size of array, the task is to find the **Kth smallest** element in the given array. It is given that all array elements are distinct.

**Expected Time Complexity:**O(n)

**Input:**  
The first line of input contains an integer **T,** denoting the number of testcases. Then T test cases follow. Each test case consists of three lines. First line of each testcase contains an integer **N** denoting size of the array. Second line contains N space separated integer denoting elements of the array. Third line of the test case contains an integer K.

**Output:**  
Corresponding to each test case, print the kth smallest element in a new line.

**Constraints:**  
1 <= T <= 100  
1 <= N <= 105  
1 <= arr[i] <= 105  
1 <= K <= N

**Example:  
Input:**  
2  
6  
7 10 4 3 20 15  
3  
5  
7 10 4 20 15  
4

**Output:**  
7  
15

**Explanation:  
Testcase 1:** 3rd smallest element in the given array is 7.

## [Trapping Rain Water](https://practice.geeksforgeeks.org/problems/trapping-rain-water/0)

## [Pythagorean Triplet](https://practice.geeksforgeeks.org/problems/pythagorean-triplet/0)

## [Chocolate Distribution Problem](https://practice.geeksforgeeks.org/problems/chocolate-distribution-problem/0)

## [Stock buy and sell](https://practice.geeksforgeeks.org/problems/stock-buy-and-sell/0)

Company Tags [Accolite](https://practice.geeksforgeeks.org/company/Accolite/)  [Amazon](https://practice.geeksforgeeks.org/company/Amazon/)  [D-E-Shaw](https://practice.geeksforgeeks.org/company/D-E-Shaw/)  [Directi](https://practice.geeksforgeeks.org/company/Directi/)  [Facebook](https://practice.geeksforgeeks.org/company/Facebook/)  [Flipkart](https://practice.geeksforgeeks.org/company/Flipkart/)  [Goldman Sachs](https://practice.geeksforgeeks.org/company/Goldman%20Sachs/)  [Hike](https://practice.geeksforgeeks.org/company/Hike/)  [Intuit](https://practice.geeksforgeeks.org/company/Intuit%20/)  [MakeMyTrip](https://practice.geeksforgeeks.org/company/MakeMyTrip/)  [Microsoft](https://practice.geeksforgeeks.org/company/Microsoft/)  [Morgan Stanley](https://practice.geeksforgeeks.org/company/Morgan%20Stanley/)  [Ola Cabs](https://practice.geeksforgeeks.org/company/Ola%20Cabs/)  [Oracle](https://practice.geeksforgeeks.org/company/Oracle/)  [Paytm](https://practice.geeksforgeeks.org/company/Paytm/)  [Pubmatic](https://practice.geeksforgeeks.org/company/Pubmatic/)  [Quikr](https://practice.geeksforgeeks.org/company/Quikr/)  [Salesforce](https://practice.geeksforgeeks.org/company/Salesforce/)  [Samsung](https://practice.geeksforgeeks.org/company/Samsung/)  [SAP Labs](https://practice.geeksforgeeks.org/company/SAP%20Labs/)  [Sapient](https://practice.geeksforgeeks.org/company/Sapient/)  [Swiggy](https://practice.geeksforgeeks.org/company/Swiggy/)  [Walmart](https://practice.geeksforgeeks.org/company/Walmart/)

* [**Problems**](https://practice.geeksforgeeks.org/problems/stock-buy-and-sell/0#problems)

The cost of stock on each day is given in an array **A[]**of size**N**. Find all the days on which you buy and sell the stock so that in between those days your profit is maximum.

**Input:**   
First line contains number of test cases **T**. First line of each test case contains an integer value **N** denoting the number of days, followed by an array of stock prices of N days.

**Output:**  
For each testcase, output all the days with profit in a single line. And if there is no profit then print "**No Profit**".

**Constraints:**  
1 <= T <= 100  
2 <= N <= 103  
0 <= Ai <= 104

**Example  
Input:**  
2  
7  
100 180 260 310 40 535 695  
10  
23 13 25 29 33 19 34 45 65 67

**Output:**  
(0 3) (4 6)  
(1 4) (5 9)

**Explanation:  
Testcase 1:**We can buy stock on day 0, and sell it on 3rd day, which will give us maximum profit.

**Note:** Output format is as follows - (buy\_day sell\_day) (buy\_day sell\_day)  
For each input, output should be in a single line.

#include <iostream>

#include <bits/stdc++.h>

using namespace std;

void buyNSell(vector<int> &A, int n){

if(n<=1) return;

bool profit=0;

int min=0, max=0;

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

if(A[i]<=A[i-1]){

if(min!=max)

cout<<"("<<min<<" "<<max<<") ";

min=max=i;

}

else{

profit=1;

max=i;

}

}

if(min!=max && profit==1)

cout<<"("<<min<<" "<<max<<")";

if(!profit)

cout<<"No Profit";

}

int main() {

//code

int t,n;

vector<int> A;

cin>>t;

while(t--){

cin>>n;

A.resize(n);

for(int i=0; i<n; i++) cin>>A[i];

buyNSell(A,n);

cout<<endl;

A.clear();

}

return 0;

}

## [Element with left side smaller and right side greater](https://practice.geeksforgeeks.org/problems/unsorted-array/0)

Company Tags [Amazon](https://practice.geeksforgeeks.org/company/Amazon/)  [Intuit](https://practice.geeksforgeeks.org/company/Intuit%20/)  [OYO Rooms](https://practice.geeksforgeeks.org/company/OYO%20Rooms/)  [Zoho](https://practice.geeksforgeeks.org/company/Zoho/)

* [**Problems**](https://practice.geeksforgeeks.org/problems/unsorted-array/0#problems)

Given an unsorted array of size **N**. Find the first element in array such that all of its left elements are smaller and all right elements to it are greater than it.  
  
**Note:** Left and right side elements can be equal to required element. And extreme elements cannot be required element.

**Input:**  
The first line of input contains an integer T denoting the number of test cases. Then T test cases follow. Each test case consists of two lines. First line of each test case contains an Integer N denoting size of array and the second line contains N space separated array elements.

**Output:**  
For each test case, in a new line print the required element. If no such element present in array then print -1.

**Constraints:**  
1 <= T <= 100  
3 <= N <= 106  
1 <= A[i] <= 106

**Example:  
Input:**  
3  
4  
4 2 5 7  
3  
11 9 12  
6  
4 3 2 7 8 9

**Output:**  
5  
-1  
7  
**Explanation:  
Testcase 1 :** Elements on left of 5 are smaller than 5 and on right of it are greater than 5.

#include <iostream>

#include <bits/stdc++.h>

using namespace std;

int findElement(vector<int> &A, int n){

if(n<3) return -1;

vector<int> Rmin(n), Lmax(n);

Lmax[0]=A[0];

Rmin[n-1]=A[n-1];

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

int j=n-1-i;

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

Rmin[j]=min(Rmin[j+1],A[j]);

}

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

if(Lmax[i]==A[i] && A[i]==Rmin[i]) return A[i];

}

return -1;

}

int main() {

//code

int t,n;

vector<int> A;

cin>>t;

while(t--){

cin>>n;

A.resize(n);

for(int i=0; i<n; i++) cin>>A[i];

cout<<findElement(A,n)<<endl;

A.clear();

}

return 0;

}

## [Convert array into Zig-Zag fashion](https://practice.geeksforgeeks.org/problems/convert-array-into-zig-zag-fashion/0)

Company Tags [Amazon](https://practice.geeksforgeeks.org/company/Amazon/)  [Paytm](https://practice.geeksforgeeks.org/company/Paytm/)

* [**Problems**](https://practice.geeksforgeeks.org/problems/convert-array-into-zig-zag-fashion/0#problems)

Given an array **A** (distinct elements) of size **N**. Rearrange the elements of array in zig-zag fashion. The converted array should be in form **a < b > c < d > e < f.** The relative order of elements is same in the output **i.e** you have to iterate on the original array only.

**Input:**  
The first line of input contains an integer **T** denoting the number of test cases.**T** testcases follow. Each testcase contains two lines of input. The first line contains a single integer **N** denoting the size of array.  
The second line contains **N** space-separated integers denoting the elements of the array.

**Output:**  
For each testcase, print the array in Zig-Zag fashion.

**Constraints:**  
1 <= T <= 100  
1 <= N <= 100  
0 <= Ai <= 10000

**Example:**  
Input:  
2  
7  
4 3 7 8 6 2 1  
4  
1 4 3 2  
Output:  
3 7 4 8 2 6 1  
1 4 2 3

A **Simple Solution** is to first sort the array. After sorting, exclude the first element, swap the remaining elements in pairs. (i.e. keep arr[0] as it is, swap arr[1] and arr[2], swap arr[3] and arr[4], and so on). Time complexity is O(nlogn) since we need to sort the array first.

We can convert in O(n) time using an **Efficient Approach**. The idea is to use modified one pass of bubble sort. Maintain a flag for representing which order(i.e. < or >) currently we need. If the current two elements are not in that order then swap those elements otherwise not.  
Let us see the main logic using three consecutive elements A, B, C. Suppose we are processing B and C currently and the current relation is ‘<'. But we have B > C. Since current relation is ‘<' previous relation must be '>‘ i.e., A must be greater than B. So, the relation is A > B and B > C. We can deduce A > C. So if we swap B and C then the relation is A > C and C < B. Finally we get the desired order **A C B.**

// Program for zig-zag conversion of array

void zigZag(int arr[], int n)

{

    // Flag true indicates relation "<" is expected,

    // else ">" is expected.  The first expected relation

    // is "<"

    bool flag = true;

    for (int i=0; i<=n-2; i++)

    {

        if (flag)  /\* "<" relation expected \*/

        {

            /\* If we have a situation like A > B > C,

               we get A > B < C by swapping B and C \*/

            if (arr[i] > arr[i+1])

                swap(arr[i], arr[i+1]);

        }

        else /\* ">" relation expected \*/

        {

            /\* If we have a situation like A < B < C,

               we get A < C > B by swapping B and C \*/

            if (arr[i] < arr[i+1])

                swap(arr[i], arr[i+1]);

        }

        flag = !flag; /\* flip flag \*/

    }

}

## [Last Index of 1](https://practice.geeksforgeeks.org/problems/last-index-of-1/0)

## [Spirally traversing a matrix](https://practice.geeksforgeeks.org/problems/spirally-traversing-a-matrix/0)

**Spirally traversing a matrix**

Company Tags [Accolite](https://practice.geeksforgeeks.org/company/Accolite/)  [Adobe](https://practice.geeksforgeeks.org/company/Adobe/)  [Amazon](https://practice.geeksforgeeks.org/company/Amazon/)  [BrowserStack](https://practice.geeksforgeeks.org/company/BrowserStack/)  [D-E-Shaw](https://practice.geeksforgeeks.org/company/D-E-Shaw/)  [FactSet](https://practice.geeksforgeeks.org/company/FactSet/)  [MakeMyTrip](https://practice.geeksforgeeks.org/company/MakeMyTrip/)  [MAQ Software](https://practice.geeksforgeeks.org/company/MAQ%20Software/)  [Microsoft](https://practice.geeksforgeeks.org/company/Microsoft/)  [Morgan Stanley](https://practice.geeksforgeeks.org/company/Morgan%20Stanley/)  [Nagarro](https://practice.geeksforgeeks.org/company/Nagarro/)  [nearbuy](https://practice.geeksforgeeks.org/company/nearbuy/)  [Oracle](https://practice.geeksforgeeks.org/company/Oracle/)  [OYO Rooms](https://practice.geeksforgeeks.org/company/OYO%20Rooms/)  [Paytm](https://practice.geeksforgeeks.org/company/Paytm/)  [Salesforce](https://practice.geeksforgeeks.org/company/Salesforce/)  [Snapdeal](https://practice.geeksforgeeks.org/company/Snapdeal/)  [Zoho](https://practice.geeksforgeeks.org/company/Zoho/)

* [**Problems**](https://practice.geeksforgeeks.org/problems/spirally-traversing-a-matrix/0#problems)

Given a matrix **mat[][]** of size **M\*N**. Traverse and print the matrix in spiral form.

**Input:**  
The first line of the input contains a single integer **T**, denoting the number of test cases. Then **T** test cases follow. Each testcase has 2 lines. First line contains **M** and **N** respectively separated by a space. Second line contains **M\*N** values separated by spaces.

**Output:**  
Elements when travelled in Spiral form, will be displayed in a single line.

**Constraints:**  
1 <= T <= 100  
2 <= M,N <= 10  
0 <= Ai <= 100

**Example:**  
**Input:**  
2  
4 4  
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16  
3 4  
1 2 3 4 5 6 7 8 9 10 11 12

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

**Explanation:**  
**Testcase 1:**  
void spiralPrint(int m, int n, int a[R][C])

{

    int i, k = 0, l = 0;

    /\* k - starting row index

        m - ending row index

        l - starting column index

        n - ending column index

        i - iterator

    \*/

    while (k < m && l < n) {

        /\* Print the first row from

               the remaining rows \*/

        for (i = l; i < n; ++i) {

            cout << a[k][i] << " ";

        }

        k++;

        /\* Print the last column

         from the remaining columns \*/

        for (i = k; i < m; ++i) {

            cout << a[i][n - 1] << " ";

        }

        n--;

        /\* Print the last row from

                the remaining rows \*/

        if (k < m) {

            for (i = n - 1; i >= l; --i) {

                cout << a[m - 1][i] << " ";

            }

            m--;

        }

        /\* Print the first column from

                   the remaining columns \*/

        if (l < n) {

            for (i = m - 1; i >= k; --i) {

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

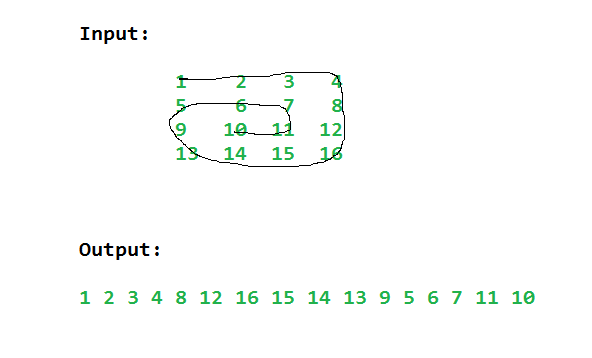
            }

            l++;

        }

    }

}



## [Largest Number formed from an Array](https://practice.geeksforgeeks.org/problems/largest-number-formed-from-an-array/0)

Company Tags [Amazon](https://practice.geeksforgeeks.org/company/Amazon/)  [MakeMyTrip](https://practice.geeksforgeeks.org/company/MakeMyTrip/)  [Microsoft](https://practice.geeksforgeeks.org/company/Microsoft/)  [Paytm](https://practice.geeksforgeeks.org/company/Paytm/)  [Zoho](https://practice.geeksforgeeks.org/company/Zoho/)

* [**Problems**](https://practice.geeksforgeeks.org/problems/largest-number-formed-from-an-array/0#problems)

Given a list of non negative integers, arrange them in such a manner that they form the largest number possible.The result is going to be very large, hence return the result in the form of a string.

**Input:**  
The first line of input consists number of the test cases. The description of T test cases is as follows:  
The first line of each test case contains the size of the array, and the second line has the elements of the array.

**Output:**  
In each separate line print the largest number formed by arranging the elements of the array **in the form of a string**.

**Constraints:**  
1 ≤ T ≤ 100  
1 ≤ N ≤ 102  
0 ≤ A[i] ≤ 103

**Example:**  
**Input:**  
2  
5  
3 30 34 5 9  
4  
54 546 548 60

**Output:**  
9534330  
6054854654

#include <iostream>

#include<bits/stdc++.h>

using namespace std;

struct myclass{

bool operator()(const int a, const int b){

return (to\_string(a)+to\_string(b) > to\_string(b)+to\_string(a));

}

}myobj;

string maxNum(vector<int> &A, int n){

string res;

if(n<1) return res;

sort(A.begin(), A.end(), myobj);

res="";

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

res+=to\_string(A[i]);

return res;

}

int main() {

//code

int t,n;

vector<int> A;

cin>>t;

while(t--){

cin>>n;

A.resize(n);

for(int i=0; i<n; i++) cin>>A[i];

cout<<maxNum(A,n)<<endl;

A.clear();

}

return 0;

}

# String

## [Reverse words in a given string](https://practice.geeksforgeeks.org/problems/reverse-words-in-a-given-string/0)

Company Tags [Accolite](https://practice.geeksforgeeks.org/company/Accolite/)  [Adobe](https://practice.geeksforgeeks.org/company/Adobe/)  [Amazon](https://practice.geeksforgeeks.org/company/Amazon/)  [Cisco](https://practice.geeksforgeeks.org/company/Cisco/)  [Goldman Sachs](https://practice.geeksforgeeks.org/company/Goldman%20Sachs/)  [MakeMyTrip](https://practice.geeksforgeeks.org/company/MakeMyTrip/)  [MAQ Software](https://practice.geeksforgeeks.org/company/MAQ%20Software/)  [Microsoft](https://practice.geeksforgeeks.org/company/Microsoft/)  [Morgan Stanley](https://practice.geeksforgeeks.org/company/Morgan%20Stanley/)  [Ola Cabs](https://practice.geeksforgeeks.org/company/Ola%20Cabs/)  [Paytm](https://practice.geeksforgeeks.org/company/Paytm/)  [Samsung](https://practice.geeksforgeeks.org/company/Samsung/)  [SAP Labs](https://practice.geeksforgeeks.org/company/SAP%20Labs/)  [Walmart](https://practice.geeksforgeeks.org/company/Walmart/)  [Wipro](https://practice.geeksforgeeks.org/company/Wipro/)  [Zoho](https://practice.geeksforgeeks.org/company/Zoho/)

* [**Problems**](https://practice.geeksforgeeks.org/problems/reverse-words-in-a-given-string/0#problems)

Given a String of length **S**, reverse the whole string without reversing the individual words in it. Words are separated by **dots**.

**Input:**  
The first line contains **T** denoting the number of testcases. T testcases follow. Each case contains a string S containing characters.

**Output:**  
For each test case, in a new line, output a single line containing the reversed String.

**Constraints:**  
1 <= T <= 100  
1 <= |S| <= 2000

**Example:**  
**Input:**  
2  
i.like.this.program.very.much  
pqr.mno

**Output:**  
much.very.program.this.like.i  
mno.pqr

Sol1:

void reverseWords(string s){

int i=0, len=s.length();

int start=0, end=0;

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

if(s[i] == '.' || i == len-1){

if(i == len-1) end++;

reverse(s.begin()+start, s.begin()+end);

start=i+1;

end=i+1;

}

else{

end++;

}

//cout<<start<<" "<<end<<endl;

}

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

cout<<s<<endl;

}

Sol2:

#include <iostream>

#include <bits/stdc++.h>

using namespace std;

int main() {

//code

string input, res="", word;

int t;

cin>>t;

while(t--){

cin>>input;

stringstream ss(input);

while(getline(ss,word,'.'))

res=word+"."+res;

cout<<res.substr(0, res.size()-1)<<endl;

res.clear();

input.clear();

word.clear();

}

return 0;

}

## [Permutations of a given string](https://practice.geeksforgeeks.org/problems/permutations-of-a-given-string/0)

Company Tags [Accolite](https://practice.geeksforgeeks.org/company/Accolite/)  [Amazon](https://practice.geeksforgeeks.org/company/Amazon/)  [Cisco](https://practice.geeksforgeeks.org/company/Cisco/)  [Citrix](https://practice.geeksforgeeks.org/company/Citrix/)  [MAQ Software](https://practice.geeksforgeeks.org/company/MAQ%20Software/)  [OYO Rooms](https://practice.geeksforgeeks.org/company/OYO%20Rooms/)  [Samsung](https://practice.geeksforgeeks.org/company/Samsung/)  [Snapdeal](https://practice.geeksforgeeks.org/company/Snapdeal/)  [Walmart](https://practice.geeksforgeeks.org/company/Walmart/)  [Zoho](https://practice.geeksforgeeks.org/company/Zoho/)

* [**Problems**](https://practice.geeksforgeeks.org/problems/permutations-of-a-given-string/0#problems)

Given a string S. The task is to print all permutations of a given string.

**Input:**  
The first line of input contains an integer **T,** denoting the number of test cases. Each test case contains a single string **S**in capital letter.

**Output:**  
For each test case, print all permutations of a given string **S**with single space and all permutations should be in lexicographically increasing order.

**Constraints:**  
1 ≤ T ≤ 10  
1 ≤ size of string ≤ 5

**Example:**  
**Input:**  
2  
ABC  
ABSG

**Output:**  
ABC ACB BAC BCA CAB CBA   
ABGS ABSG AGBS AGSB ASBG ASGB BAGS BASG BGAS BGSA BSAG BSGA GABS GASB GBAS GBSA GSAB GSBA SABG SAGB SBAG SBGA SGAB SGBA

**Explanation:  
Testcase 1:** Given string ABC has permutations in 6 forms as ABC, ACB, BAC, BCA, CAB and CBA .

## [Longest Palindrome in a String](https://practice.geeksforgeeks.org/problems/longest-palindrome-in-a-string/0)

Company Tags [Accolite](https://practice.geeksforgeeks.org/company/Accolite/)  [Amazon](https://practice.geeksforgeeks.org/company/Amazon/)  [Groupon](https://practice.geeksforgeeks.org/company/Groupon/)  [MakeMyTrip](https://practice.geeksforgeeks.org/company/MakeMyTrip/)  [Microsoft](https://practice.geeksforgeeks.org/company/Microsoft/)  [Qualcomm](https://practice.geeksforgeeks.org/company/Qualcomm/)  [Samsung](https://practice.geeksforgeeks.org/company/Samsung/)  [Visa](https://practice.geeksforgeeks.org/company/Visa/)  [Walmart](https://practice.geeksforgeeks.org/company/Walmart/)  [Zoho](https://practice.geeksforgeeks.org/company/Zoho/)

* [**Problems**](https://practice.geeksforgeeks.org/problems/longest-palindrome-in-a-string/0#problems)

Given a string S, find the longest palindromic substring in S.**Substring of string S:** S[ i . . . . j ] where 0 ≤ i ≤ j < len(S)**. Palindrome string:** A string which reads the same backwards. More formally, S is palindrome if reverse(S) = S.**Incase of conflict**, return the substring which occurs first ( with the least starting index ).

**NOTE:** Required Time Complexity **O(n2).**

**Input:**  
The first line of input consists number of the testcases. The following **T** lines consist of a string each.

**Output:**  
In each separate line print the longest palindrome of the string given in the respective test case.

**Constraints:**  
1 ≤ T ≤ 100  
1 ≤ Str Length ≤ 104

**Example:  
Input:**  
1  
aaaabbaa

**Output:**  
aabbaa

**Explanation:  
Testcase 1:** The longest palindrome string present in the given string is "aabbaa".

## [Recursively remove all adjacent duplicates](https://practice.geeksforgeeks.org/problems/recursively-remove-all-adjacent-duplicates/0)

## [Check if string is rotated by two places](https://practice.geeksforgeeks.org/problems/check-if-string-is-rotated-by-two-places/0)

## [Roman Number to Integer](https://practice.geeksforgeeks.org/problems/roman-number-to-integer/0)

Company Tags [Amazon](https://practice.geeksforgeeks.org/company/Amazon/)  [Facebook](https://practice.geeksforgeeks.org/company/Facebook/)  [Microsoft](https://practice.geeksforgeeks.org/company/Microsoft/)  [Twitter](https://practice.geeksforgeeks.org/company/Twitter/)  [Zoho](https://practice.geeksforgeeks.org/company/Zoho/)

* [**Problems**](https://practice.geeksforgeeks.org/problems/roman-number-to-integer/0#problems)

Given an string in roman no format (s)  your task is to convert it to integer .  
  
**Input:**  
The first line of each test case contains the no of test cases T. Then T test cases follow. Each test case contains a string s denoting the roman no.  
  
**Output:**  
For each test case in a new line print the integer representation of roman number s.   
  
**Constraints:**  
1<=T<=100  
1<=roman no range<4000  
  
**Example:  
Input**  
2  
V  
III   
**Output**  
5  
3

int getVal(char c){

int val=-1;

switch(c){

case 'I': val=1; break;

case 'V': val=5; break;

case 'X': val=10; break;

case 'L': val=50; break;

case 'C': val=100; break;

case 'D': val=500; break;

case 'M': val=1000; break;

}

return val;

}

int Solution::romanToInt(string A) {

int res=0;

int n=A.size();

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

int v1=getVal(A[i]);

if(i+1<n){

int v2=getVal(A[i+1]);

if(v1>=v2)

res+=v1;

else{

res+=v2-v1;

i++;

}

}

else

res+=v1;

}

return res;

}

## [Anagram](https://practice.geeksforgeeks.org/problems/anagram/0)

Company Tags [Amazon](https://practice.geeksforgeeks.org/company/Amazon/)  [Goldman Sachs](https://practice.geeksforgeeks.org/company/Goldman%20Sachs/)  [Nagarro](https://practice.geeksforgeeks.org/company/Nagarro/)

* [**Problems**](https://practice.geeksforgeeks.org/problems/anagram/0#problems)

Given two strings **a** and **b** consisting of lowercase characters. The task is to check whether two given strings are anagram of each other or not. An anagram of a string is another string that contains same characters, only the order of characters can be different. For example, “act” and “tac” are anagram of each other.

**Input:**  
The first line of input contains an integer T denoting the number of test cases. Each test case consist of two strings in 'lowercase' only, in a single line.

**Output:**  
Print "**YES**" without quotes if the two strings are anagram else print "**NO**".

**Constraints:**  
1 ≤ T ≤ 300  
1 ≤ |s| ≤ 1016

**Example:**  
**Input:**  
2  
geeksforgeeks forgeeksgeeks  
allergy allergic

**Output:**  
YES  
NO

**Explanation:  
Testcase 1:** Both the string have same characters with same frequency. So, both are anagrams.  
**Testcase 2:** Characters in both the strings are not same, so they are not anagrams.

#include <iostream>

#include <bits/stdc++.h>

using namespace std;

bool anagram(string s1, string s2){

if(s1.size()!=s2.size())

return false;

vector<int> chars(26,0);

for(int i=0; i<s1.size(); i++)

chars[s1[i]-'a']++;

for(int i=0; i<s2.size(); i++)

chars[s2[i]-'a']--;

int sum=accumulate(chars.begin(), chars.end(),0);

if(sum!=0) return false;

return true;

}

int main() {

//code

int t;

string s1, s2;

cin>>t;

while(t--){

cin>>s1>>s2;

if(anagram(s1, s2)) cout<<"YES"<<endl;

else cout<<"NO"<<endl;

}

return 0;

}

## [Remove Duplicates](https://practice.geeksforgeeks.org/problems/remove-duplicates/0)

Company Tags [Microsoft](https://practice.geeksforgeeks.org/company/Microsoft/)

* [**Problems**](https://practice.geeksforgeeks.org/problems/remove-duplicates/0#problems)

Given a **string**, the task is to remove duplicates from it. Expected time complexity O(n) where n is length of input string and extra space O(1) under the assumption that there are total 256 possible characters in a string.

**Note:** that original order of characters must be kept same.

**Input:**  
First line of the input is the number of test cases **T**. And first line of test case contains a string.

**Output:**  
Modified string without duplicates and same order of characters.

**Constraints:**   
1 <= T <= 15  
1 <= |string|<= 1000  
  
**Example:**  
**Input:**  
2  
geeksforgeeks  
geeks for geeks

**Output:**  
geksfor  
geks for

## [Form a Palindrome](https://practice.geeksforgeeks.org/problems/form-a-palindrome/0)

Company Tags [Airtel](https://practice.geeksforgeeks.org/company/Airtel/)  [Amazon](https://practice.geeksforgeeks.org/company/Amazon/)  [Google](https://practice.geeksforgeeks.org/company/Google/)

* [**Problems**](https://practice.geeksforgeeks.org/problems/form-a-palindrome/0#problems)

Given a string, find the minimum number of characters to be inserted to convert it to palindrome.  
For Example:  
ab: Number of insertions required is 1. **b**ab or aba  
aa: Number of insertions required is 0. aa  
abcd: Number of insertions required is 3. **dcb**abcd

**Input:**

The first line of input contains an integer T denoting the number of test cases.  
The first line of each test case is S.  
  
**Output:**

Print the minimum number of characters.  
  
**Constraints:**

1 ≤ T ≤ 50  
1 ≤ S ≤ 40  
  
**Example:**

**Input:**  
3  
abcd  
aba  
geeks

**Output:**  
3  
0  
3

#### Sol1:

 If we find out LCS of string and its reverse, we know how many maximum characters can form a palindrome. We need insert remaining characters. Following are the steps.

1. Find the length of LCS of input string and its reverse. Let the length be ‘l’.
2. The minimum number insertions needed is length of input string minus ‘l’.

#include <iostream>

#include<bits/stdc++.h>

using namespace std;

int lcs(string A, string B){

int lenA=A.size();

int lenB=B.size();

if(lenA==0 || lenB==0) return 0;

vector<vector<int>>dp(lenA+1, vector<int>(lenB+1,0));

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

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

if(i==0 || j==0) continue;

if(A[i-1]==B[j-1])

dp[i][j]=dp[i-1][j-1]+1;

else

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

}

}

return dp[lenA][lenB];

}

int countChar(string s){

int len=s.size();

if(len <= 1) return 0;

string rs=s;

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

return len-lcs(s,rs);

}

int main() {

//code

int t;

cin>>t;

while(t--){

string s;

cin>>s;

cout<<countChar(s)<<endl;

}

return 0;

}

#### Sol2:

#include <iostream>

#include<bits/stdc++.h>

using namespace std;

int countChar(string s){

int n=s.size();

if(n<=1) return 0;

vector<vector<int>>dp(n,vector<int>(n,0));

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

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

int j=i+d;

if(s[i]==s[j]) dp[i][j]=dp[i+1][j-1];

else{

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

}

}

}

return dp[0][n-1];

}

int main() {

//code

int t;

cin>>t;

while(t--){

string s;

cin>>s;

cout<<countChar(s)<<endl;

}

return 0;

}

## [Longest Distinct Characters in the string](https://practice.geeksforgeeks.org/problems/longest-distinct-characters-in-string/0)

Company Tags [Adobe](https://practice.geeksforgeeks.org/company/Adobe/)  [Amazon](https://practice.geeksforgeeks.org/company/Amazon/)  [Housing.com](https://practice.geeksforgeeks.org/company/Housing.com/)  [Microsoft](https://practice.geeksforgeeks.org/company/Microsoft/)

* [**Problems**](https://practice.geeksforgeeks.org/problems/longest-distinct-characters-in-string/0#problems)

Given a string **S**, find length of the longest substring with all distinct characters.  For example, for input "abca", the output is 3 as "abc" is the longest substring with all distinct characters.

**Input:**  
The first line of input contains an integer T denoting the number of test cases.  
The first line of each test case is String str.

**Output:**  
Print length of smallest substring with maximum number of distinct characters.  
**Note:** The output substring should have all distinct characters.

**Constraints:**  
1 ≤ T ≤ 100  
1 ≤ size of str ≤ 10000

**Example:**  
**Input:**  
2  
abababcdefababcdab  
geeksforgeeks

**Output:**  
6  
7

#### Approach:

We start traversing the string from left to right and maintain track of:

1. the current substring with non-repeating characters with the help of a *start* and *end* index
2. the longest non-repeating substring *res*
3. **a lookup map of already *visited* characters**

#include <iostream>

#include <bits/stdc++.h>

using namespace std;

int NRCS(string A){

string res;

map<char,int>m;

for(int start=0, end=0; end<A.size(); end++){

char curChar=A.at(end);

if(m.find(curChar)!=m.end())

start=max(start, m[curChar]+1);

if(res.size()<end-start+1)

res=A.substr(start, end-start+1);

m[curChar]=end;

}

return res.size();

}

int main() {

//code

int t;

string s;

cin>>t;

while(t--){

cin>>s;

cout<<NRCS(s)<<endl;

}

return 0;

}

## [Implement Atoi](https://practice.geeksforgeeks.org/problems/implement-atoi/1)

Company Tags [Adobe](https://practice.geeksforgeeks.org/company/Adobe/)  [Amazon](https://practice.geeksforgeeks.org/company/Amazon/)  [Code Brew](https://practice.geeksforgeeks.org/company/Code%20Brew/)  [Microsoft](https://practice.geeksforgeeks.org/company/Microsoft/)  [Morgan Stanley](https://practice.geeksforgeeks.org/company/Morgan%20Stanley/)  [Payu](https://practice.geeksforgeeks.org/company/Payu/)

* [**Problems**](https://practice.geeksforgeeks.org/problems/implement-atoi/1#problems)

Your task  is to implement the function **atoi**. The function takes a string(str) as argument and converts it to an integer and returns it.  
  
**Input:**  
The first line of input contains an integer T denoting the no of test cases . Then T test cases follow. Each test case contains a string str .  
  
**Output:**  
For each test case in a new line output will be an integer denoting the converted integer, if the input string is not a numerical string then output will be -1.  
  
**Constraints:**  
1<=T<=100  
1<=length of (s,x)<=10  
  
**Example(To be used only for expected output) :  
Input:**  
2  
123  
21a  
  
**Output:**  
123  
-1

int atoi(string A)

{

//Your code here

long double res=0;

int sign=1;

for(int i=0; i<A.size(); i++){

if(i==0 && A[i]=='-'){

sign=-1;

continue;

}

if(!isdigit(A[i])) return -1;

res = res\*10 + (A[i]-'0');

}

return res\*sign;

}

## [Implement strstr](https://practice.geeksforgeeks.org/problems/implement-strstr/1)

Company Tags [Amazon](https://practice.geeksforgeeks.org/company/Amazon/)  [Facebook](https://practice.geeksforgeeks.org/company/Facebook/)  [Microsoft](https://practice.geeksforgeeks.org/company/Microsoft/)  [Qualcomm](https://practice.geeksforgeeks.org/company/Qualcomm/)

* [**Problems**](https://practice.geeksforgeeks.org/problems/implement-strstr/1#problems)

Your task is to implement the function **strstr**. The function takes two strings as arguments **(s,x)** and  locates the occurrence of the string**x** in the string **s**. The function returns and integer denoting the **first occurrence**of the string x in s.

**Input Format:**  
The first line of input contains an integer **T** denoting the no of test cases . Then T test cases follow. The first line of each test case contains two strings **s and x**.

**Output Format:**  
For each test case, in a new line, output will be an integer denoting the first occurrence of the x in the string s. Return**-1** if no match found.

**Your Task:**  
Since this is a function problem, you don't have to take any input. Just complete the **strstr()**function. The function returns -1 if no match if found else it returns an integer denoting the first occurrence of the x in the string s.

**Constraints:**  
1 <= T <= 200  
1<= |s|,|x| <= 1000

**Example:  
Input**  
2  
GeeksForGeeks Fr  
GeeksForGeeks For  
**Output**  
-1  
5

**Explanation:**  
**Testcase 1:** Fr is not present in the string GeeksForGeeks as substring.  
**Testcase 2:** For is present as substring in GeeksForGeeks from index 5.

int strstr(string s, string x)

{

//Your code here

int m=s.size();

int n=x.size();

if(m==0 || n==0 || n>m) return -1;

int i=0, j=0, start=0;

while(i<m && j<n && start<=m-n){

if(s[i]==x[j]){

if(j==n-1) return start;

i++;

j++;

}

else{

j=0;

i=++start;

}

}

## [Longest Common Prefix](https://practice.geeksforgeeks.org/problems/longest-common-prefix-in-an-array/0)

Company Tags [VMWare](https://practice.geeksforgeeks.org/company/VMWare/)

* [**Problems**](https://practice.geeksforgeeks.org/problems/longest-common-prefix-in-an-array/0#problems)

Given a array of **N** strings, find the longest common prefix among all strings present in the array.

**Input:**  
The first line of the input contains an integer **T** which denotes the number of test cases to follow. Each test case contains an integer **N**. Next line has space separated **N** strings.

**Output:**  
Print the longest common prefix as a string in the given array. If no such prefix exists print "-1"(without quotes).  
  
**Constraints:**  
1 <= T <= 103  
1 <= N <= 103  
1 <= |S| <= 103

**Example:  
Input:**  
2  
4  
geeksforgeeks geeks geek geezer  
3  
apple ape april

**Output:**  
gee  
ap

#### Sol:

bool isSame(vector<string> &A, int i){

char c=A[0][i];

for(int j=1; j<A.size(); j++)

if(c!=A[j][i])

return false;

return true;

}

string Solution::longestCommonPrefix(vector<string> &A) {

string res;

int n=A.size();

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

string first=A[0];

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

minLen=min(minLen, (int)A[i].size());

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

if(isSame(A,i))

res.push\_back(first[i]);

else

break;

}

return res;

}

# **Linked List**

## [Finding middle element in a linked list](https://practice.geeksforgeeks.org/problems/finding-middle-element-in-a-linked-list/1)

Company Tags [Adobe](https://practice.geeksforgeeks.org/company/Adobe/)  [Amazon](https://practice.geeksforgeeks.org/company/Amazon/)  [Flipkart](https://practice.geeksforgeeks.org/company/Flipkart/)  [GE](https://practice.geeksforgeeks.org/company/GE/)  [Hike](https://practice.geeksforgeeks.org/company/Hike/)  [IgniteWorld](https://practice.geeksforgeeks.org/company/IgniteWorld%20/)  [MAQ Software](https://practice.geeksforgeeks.org/company/MAQ%20Software/)  [Microsoft](https://practice.geeksforgeeks.org/company/Microsoft/)  [Morgan Stanley](https://practice.geeksforgeeks.org/company/Morgan%20Stanley/)  [Nagarro](https://practice.geeksforgeeks.org/company/Nagarro/)  [Netskope](https://practice.geeksforgeeks.org/company/Netskope%20/)  [Payu](https://practice.geeksforgeeks.org/company/Payu/)  [Qualcomm](https://practice.geeksforgeeks.org/company/Qualcomm/)  [Samsung](https://practice.geeksforgeeks.org/company/Samsung/)  [SAP Labs](https://practice.geeksforgeeks.org/company/SAP%20Labs/)  [Veritas](https://practice.geeksforgeeks.org/company/Veritas/)  [VMWare](https://practice.geeksforgeeks.org/company/VMWare/)  [Wipro](https://practice.geeksforgeeks.org/company/Wipro/)  [Zoho](https://practice.geeksforgeeks.org/company/Zoho/)

* [**Problems**](https://practice.geeksforgeeks.org/problems/finding-middle-element-in-a-linked-list/1#problems)

Given a singly linked list of **N** nodes. The task is to find middle of the linked list. For example, if given linked list is 1->2->3->4->5 then output should be 3.  
If there are even nodes, then there would be two middle nodes, we need to print second middle element. For example, if given linked list is 1->2->3->4->5->6 then output should be 4.

**Input:**  
First line of input contains number of testcases T. For each testcase, first line of input contains length of linked list and next line contains data of nodes of linked list.

**Output:**  
For each testcase, there will be a single line of output containing data of middle element of linked list.

**User Task:**  
The task is to complete the function **getMiddle**() which takes head reference as the only argument and should return the data at the middle node of linked list.

**Constraints:**  
1 <= T <= 100  
1 <= N <= 100

**Example:  
Input:**  
2  
5  
1 2 3 4 5  
6  
2 4 6 7 5 1

**Output:**  
3  
7

int getMiddle(Node \*head)

{

// Your code here

if(!head) return -1;

Node \*fast=head, \*slow=head;

while(fast && fast->next){

slow=slow->next;

fast=fast->next->next;

}

return slow->data;

}

## [Reverse a linked list](https://practice.geeksforgeeks.org/problems/reverse-a-linked-list/1)

## [Rotate a Linked List](https://practice.geeksforgeeks.org/problems/rotate-a-linked-list/1)

## [Reverse a Linked List in groups of given size](https://practice.geeksforgeeks.org/problems/reverse-a-linked-list-in-groups-of-given-size/1)

## [Intersection point in Y shaped linked lists](https://practice.geeksforgeeks.org/problems/intersection-point-in-y-shapped-linked-lists/1/)

## [Detect Loop in linked list](https://practice.geeksforgeeks.org/problems/detect-loop-in-linked-list/1)

## [Remove loop in Linked List](https://practice.geeksforgeeks.org/problems/remove-loop-in-linked-list/1)

## [n’th node from end of linked list](https://practice.geeksforgeeks.org/problems/nth-node-from-end-of-linked-list/1)

## [Flattening a Linked List](https://practice.geeksforgeeks.org/problems/flattening-a-linked-list/1)

## [Merge two sorted linked lists](https://practice.geeksforgeeks.org/problems/merge-two-sorted-linked-lists/1)

## [Intersection point of two Linked Lists](https://practice.geeksforgeeks.org/problems/intersection-point-in-y-shapped-linked-lists/1)

## [Pairwise swap of a linked list](https://practice.geeksforgeeks.org/problems/pairwise-swap-elements-of-a-linked-list-by-swapping-data/1)

## [Add two numbers represented by linked lists](https://practice.geeksforgeeks.org/problems/add-two-numbers-represented-by-linked-lists/1)

## [Check if Linked List is Palindrome](https://practice.geeksforgeeks.org/problems/check-if-linked-list-is-pallindrome/1)

## [Implement Queue using Linked List](https://practice.geeksforgeeks.org/problems/implement-queue-using-linked-list/1)

## [Implement Stack using Linked List](https://practice.geeksforgeeks.org/problems/implement-stack-using-linked-list/1)

## [Given a linked list of 0s, 1s and 2s, sort it](https://practice.geeksforgeeks.org/problems/given-a-linked-list-of-0s-1s-and-2s-sort-it/1)

## [Delete without head pointer](https://practice.geeksforgeeks.org/problems/delete-without-head-pointer/1)

# **Stack and Queue**

## [Parenthesis Checker](https://practice.geeksforgeeks.org/problems/parenthesis-checker/0)

Company Tags [Adobe](https://practice.geeksforgeeks.org/company/Adobe/)  [Amazon](https://practice.geeksforgeeks.org/company/Amazon/)  [Flipkart](https://practice.geeksforgeeks.org/company/Flipkart/)  [Hike](https://practice.geeksforgeeks.org/company/Hike/)  [Oracle](https://practice.geeksforgeeks.org/company/Oracle/)  [OYO Rooms](https://practice.geeksforgeeks.org/company/OYO%20Rooms/)  [Snapdeal](https://practice.geeksforgeeks.org/company/Snapdeal/)  [Walmart](https://practice.geeksforgeeks.org/company/Walmart/)  [Wipro](https://practice.geeksforgeeks.org/company/Wipro/)  [Yatra.com](https://practice.geeksforgeeks.org/company/Yatra.com/)  [Zoho](https://practice.geeksforgeeks.org/company/Zoho/)

* [**Problems**](https://practice.geeksforgeeks.org/problems/parenthesis-checker/0#problems)

Given an expression string **exp**. Examine whether the pairs and the orders of “{“,”}”,”(“,”)”,”[“,”]” are correct in exp.  
For example, the program should print 'balanced' for exp = “[()]{}{[()()]()}” and 'not balanced' for exp = “[(])”

**Input:**  
The first line of input contains an integer T denoting the number of test cases.  Each test case consists of a string of expression, in a separate line.

**Output:**  
Print 'balanced' without quotes if the pair of parenthesis is balanced else print 'not balanced' in a separate line.

**Constraints:**  
1 ≤ T ≤ 100  
1 ≤ |s| ≤ 105

**Example:  
Input:**  
3  
{([])}  
()  
([]

**Output:**  
balanced  
balanced  
not balanced

int Solution::isValid(string s) {

stack<char> st;

map<char, char> matching;

matching['(']=')';

matching['{']='}';

matching['[']=']';

for(int i=0; i<s.size(); i++){

if(s[i]=='(' || s[i]=='[' || s[i]=='{')

st.push(s[i]);

else if(st.empty() || matching[st.top()]!=s[i])

return 0;

else st.pop();

}

return st.empty();

}

## [Next larger element](https://practice.geeksforgeeks.org/problems/next-larger-element/0)

## [Queue using two Stacks](https://practice.geeksforgeeks.org/problems/queue-using-two-stacks/1)

## [Stack using two queues](https://practice.geeksforgeeks.org/problems/stack-using-two-queues/1)

## [Get minimum element from stack](https://practice.geeksforgeeks.org/problems/get-minimum-element-from-stack/1)

## [LRU Cache](https://practice.geeksforgeeks.org/problems/lru-cache/1)

## [Circular tour](https://practice.geeksforgeeks.org/problems/circular-tour/1)

## [First non-repeating character in a stream](https://practice.geeksforgeeks.org/problems/first-non-repeating-character-in-a-stream/0)

## [Rotten Oranges](https://practice.geeksforgeeks.org/problems/rotten-oranges/0)

## [Maximum of all subarrays of size k](https://practice.geeksforgeeks.org/problems/maximum-of-all-subarrays-of-size-k/0)

# **Tree**

## [Print Left View of Binary Tree](https://practice.geeksforgeeks.org/problems/left-view-of-binary-tree/1)

Please note that it's Function problem i.e.

you need to write your solution in the form of Function(s) only.

Driver Code to call/invoke your function would be added by GfG's Online Judge.\*/

/\* A binary tree node

struct Node

{

int data;

Node\* left, \* right;

}; \*/

// A wrapper over leftViewUtil()

void leftViewUtil(Node \*root, int \*max\_level, int level){

if(!root) return;

if(level > \*max\_level){

cout<<root->data<<" ";

\*max\_level = level;

}

leftViewUtil(root->left, max\_level, level+1);

leftViewUtil(root->right, max\_level, level+1);

}

void leftView(Node \*root)

{

// Your code here

int level, max\_level=0;

leftViewUtil(root, &max\_level, 1);

}

## [Check for BST](https://practice.geeksforgeeks.org/problems/check-for-bst/1)

int isBSTUtil(node\* node, int min, int max);

/\* Returns true if the given

tree is a binary search tree

(efficient version). \*/

int isBST(node\* node)

{

    return(isBSTUtil(node, INT\_MIN, INT\_MAX));

}

/\* Returns true if the given

tree is a BST and its values

are >= min and <= max. \*/

int isBSTUtil(node\* node, int min, int max)

{

    /\* an empty tree is BST \*/

    if (node==NULL)

        return 1;

    /\* false if this node violates

    the min/max constraint \*/

    if (node->data < min || node->data > max)

        return 0;

    /\* otherwise check the subtrees recursively,

    tightening the min or max constraint \*/

    return

        isBSTUtil(node->left, min, node->data-1) && // Allow only distinct //values

        isBSTUtil(node->right, node->data+1, max); // Allow only distinct //values

}

## [Print Bottom View of Binary Tree](https://practice.geeksforgeeks.org/problems/bottom-view-of-binary-tree/1)

## [Print a Binary Tree in Vertical Order](https://practice.geeksforgeeks.org/problems/print-a-binary-tree-in-vertical-order/1)

## [Level order traversal in spiral form](https://practice.geeksforgeeks.org/problems/level-order-traversal-in-spiral-form/1)

## [Connect Nodes at Same Level](https://practice.geeksforgeeks.org/problems/connect-nodes-at-same-level/1)

## [Lowest Common Ancestor in a BST](https://practice.geeksforgeeks.org/problems/lowest-common-ancestor-in-a-bst/1)

## [Convert a given Binary Tree to Doubly Linked List](https://practice.geeksforgeeks.org/problems/binary-tree-to-dll/1)

## [Write Code to Determine if Two Trees are Identical or Not](https://practice.geeksforgeeks.org/problems/determine-if-two-trees-are-identical/1)

## [Given a binary tree, check whether it is a mirror of itself](https://practice.geeksforgeeks.org/problems/symmetric-tree/1)

## [Height of Binary Tree](https://practice.geeksforgeeks.org/problems/height-of-binary-tree/1)

## [Maximum Path Sum](https://practice.geeksforgeeks.org/problems/maximum-path-sum/1)

## [Diameter of a Binary Tree](https://practice.geeksforgeeks.org/problems/diameter-of-binary-tree/1)

## [Number of leaf nodes](https://practice.geeksforgeeks.org/problems/count-leaves-in-binary-tree/1)

## [Check if given Binary Tree is Height Balanced or Not](https://practice.geeksforgeeks.org/problems/check-for-balanced-tree/1)

## [Serialize and Deserialize a Binary Tree](https://practice.geeksforgeeks.org/problems/serialize-and-deserialize-a-binary-tree/1)

# **Heap**

## [Find median in a stream](https://practice.geeksforgeeks.org/problems/find-median-in-a-stream/0)

## [Heap Sort](https://practice.geeksforgeeks.org/problems/heap-sort/1)

## [Operations on Binary Min Heap](https://practice.geeksforgeeks.org/problems/operations-on-binary-min-heap/1)

## [Rearrange characters](https://practice.geeksforgeeks.org/problems/rearrange-characters/0)

## [Kth largest element in a stream](https://practice.geeksforgeeks.org/problems/kth-largest-element-in-a-stream/0)

## [Merge K sorted linked lists](https://practice.geeksforgeeks.org/problems/merge-k-sorted-linked-lists/1)

## [Kth largest element in a stream](https://practice.geeksforgeeks.org/problems/kth-largest-element-in-a-stream/0)

# **Recursion**

1. [Flood fill Algorithm](https://practice.geeksforgeeks.org/problems/flood-fill-algorithm/0)

##### **Difficulty:**[**Easy**](https://practice.geeksforgeeks.org/Easy/0/0/)**Marks: 2**

Company Tags [Google](https://practice.geeksforgeeks.org/company/Google/)

* [**Problems**](https://practice.geeksforgeeks.org/problems/flood-fill-algorithm/0#problems)

Given a 2D screen, location of a pixel in the screen ie(x,y) and a color(K), your task is to replace color of the given pixel and all adjacent(excluding diagonally adjacent) same colored pixels with the given color K.

Example:

{{1, 1, 1, 1, 1, 1, 1, 1},  
{1, 1, 1, 1, 1, 1, 0, 0},  
{1, 0, 0, 1, 1, 0, 1, 1},  
{1, **2, 2, 2, 2,** 0, 1, 0},  
{1, 1, 1,**2, 2**, 0, 1, 0},  
{1, 1, 1, **2, 2, 2, 2**, 0},  
{1, 1, 1, 1, 1, **2**, 1, 1},  
{1, 1, 1, 1, 1, **2, 2,** 1},  
 };

 x=4, y=4, color=3

{{1, 1, 1, 1, 1, 1, 1, 1},  
{1, 1, 1, 1, 1, 1, 0, 0},  
{1, 0, 0, 1, 1, 0, 1, 1},   
{1, **3, 3, 3, 3**, 0, 1, 0},  
{1, 1, 1, **3, 3**, 0, 1, 0},  
{1, 1, 1, **3, 3, 3, 3,** 0},  
{1, 1, 1, 1, 1, **3**, 1, 1},  
{1, 1, 1, 1, 1, **3, 3**, 1}, };

**Note:**Use zero based indexing.

**Input:**  
The first line of input contains an integer T denoting the no of test cases. Then T test cases follow. The first line of each test case contains Two integers N and M denoting the size of the matrix. Then in the next line are N\*M space separated values of the matrix. Then in the next line are three values x, y and K.

**Output:**  
For each test case print the space separated values of the new matrix.

**Constraints:**  
1 <= T <= 100  
1 <= M[][] <= 100

**Example:  
Input:**  
3  
3 4  
0 1 1 0 1 1 1 1 0 1 2 3  
0 1 5  
2 2  
1 1 1 1  
0 1 8  
4 4   
1 2 3 4 1 2 3 4 1 2 3 4 1 3 2 4  
0 2 9

**Output:**  
0 5 5 0 5 5 5 5 0 5 2 3  
8 8 8 8  
1 2 9 4 1 2 9 4 1 2 9 4 1 3 2 4

1. [Number of paths](https://practice.geeksforgeeks.org/problems/number-of-paths/0)
2. [Combination Sum – Part 2](https://practice.geeksforgeeks.org/problems/combination-sum-part-2/0)
3. [Special Keyboard](https://practice.geeksforgeeks.org/problems/special-keyboard/0)
4. [Josephus problem](https://practice.geeksforgeeks.org/problems/josephus-problem/1)

# **Hashing**

1. [Relative Sorting](https://practice.geeksforgeeks.org/problems/relative-sorting/0)
2. [Sorting Elements of an Array by Frequency](https://practice.geeksforgeeks.org/problems/sorting-elements-of-an-array-by-frequency/0)
3. [Largest subarray with 0 sum](https://practice.geeksforgeeks.org/problems/largest-subarray-with-0-sum/1)
4. [Common elements](https://practice.geeksforgeeks.org/problems/common-elements/0)
5. [Find all four sum numbers](https://practice.geeksforgeeks.org/problems/find-all-four-sum-numbers/0)
6. [Swapping pairs make sum equal](https://practice.geeksforgeeks.org/problems/swapping-pairs-make-sum-equal/0)
7. [Count distinct elements in every window](https://practice.geeksforgeeks.org/problems/count-distinct-elements-in-every-window/1)
8. [Array Pair Sum Divisibility Problem](https://practice.geeksforgeeks.org/problems/array-pair-sum-divisibility-problem/0)
9. [Longest consecutive subsequence](https://practice.geeksforgeeks.org/problems/longest-consecutive-subsequence/0)
10. [Array Subset of another array](https://practice.geeksforgeeks.org/problems/array-subset-of-another-array/0)
11. [Find all pairs with a given sum](https://practice.geeksforgeeks.org/problems/find-all-pairs-whose-sum-is-x/0)
12. [Find first repeated character](https://practice.geeksforgeeks.org/problems/find-first-repeated-character/0)
13. [Zero Sum Subarrays](https://practice.geeksforgeeks.org/problems/zero-sum-subarrays/0)
14. [Minimum indexed character](https://practice.geeksforgeeks.org/problems/minimum-indexed-character/0)
15. [Check if two arrays are equal or not](https://practice.geeksforgeeks.org/problems/check-if-two-arrays-are-equal-or-not/0)
16. [Uncommon characters](https://practice.geeksforgeeks.org/problems/uncommon-characters/0)
17. [Smallest window in a string containing all the characters of another string](https://practice.geeksforgeeks.org/problems/smallest-window-in-a-string-containing-all-the-characters-of-another-string/0)
18. [First element to occur k times](https://practice.geeksforgeeks.org/problems/first-element-to-occur-k-times/0)
19. [Check if frequencies can be equal](https://practice.geeksforgeeks.org/problems/check-frequencies/0)

# Graph

## [Depth First Traversal](https://practice.geeksforgeeks.org/problems/depth-first-traversal-for-a-graph/1)

## [Breadth First Traversal](https://practice.geeksforgeeks.org/problems/bfs-traversal-of-graph/1)

## [Detect cycle in undirected graph](https://practice.geeksforgeeks.org/problems/detect-cycle-in-an-undirected-graph/1/)

## [Detect cycle in a directed graph](https://practice.geeksforgeeks.org/problems/detect-cycle-in-a-directed-graph/1)

## [Topological sort](https://practice.geeksforgeeks.org/problems/topological-sort/1)

## [Find the number of islands](https://practice.geeksforgeeks.org/problems/find-the-number-of-islands/1)

[**Number of Islands**](https://leetcode.com/problems/number-of-islands/) **(Leetcode)**

Medium

3265117FavoriteShare

Given a 2d grid map of '1's (land) and '0's (water), count the number of islands. An island is surrounded by water and is formed by connecting adjacent lands horizontally or vertically. You may assume all four edges of the grid are all surrounded by water.

**Example 1:**

**Input:**

11110

11010

11000

00000

**Output:** 1

**Example 2:**

**Input:**

11000

11000

00100

00011

**Output:** 3

**Sol:**

bool isSafe(int i, int j, int R, int C){

return (i>=0 && i<R && j>=0 && j<C) ;

}

void DFSUtil(vector<vector<char>>& grid, vector<vector<bool>>& visited, int R, int C, int i, int j){

visited[i][j]=1;

if(isSafe(i+1,j,R,C) && grid[i+1][j]=='1' && !visited[i+1][j])

DFSUtil(grid, visited, R, C, i+1, j);

if(isSafe(i,j+1,R,C) && grid[i][j+1]=='1' && !visited[i][j+1])

DFSUtil(grid, visited, R, C, i, j+1);

if(isSafe(i-1,j,R,C) && grid[i-1][j]=='1' && !visited[i-1][j])

DFSUtil(grid, visited, R, C, i-1, j);

if(isSafe(i,j-1,R,C) && grid[i][j-1]=='1' && !visited[i][j-1])

DFSUtil(grid, visited, R, C, i, j-1);

}

class Solution {

public:

int numIslands(vector<vector<char>>& grid) {

int islands=0;

int R=grid.size();

if(R<1) return islands;

int C=grid[0].size();

if(C<1) return islands;

vector<vector<bool>> visited(R, vector<bool>(C,false));

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

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

if(grid[i][j]=='1' && !visited[i][j]){

//cout<<i<<" "<<j<<" "<<visited[i][j]<<endl;

DFSUtil(grid, visited, R, C, i, j);

islands++;

}

}

}

return islands;

}

};

## [Implementing Dijkstra](https://practice.geeksforgeeks.org/problems/implementing-dijkstra-set-1-adjacency-matrix/1)

## [Minimum Swaps](https://practice.geeksforgeeks.org/problems/minimum-swaps/1)

int minSwaps(vector<int> &A){

int n=A.size();

if(n<=1) return 0;

vector<pair<int,int>> B(n);

vector<bool> visited(n, false);

int res=0;

sort(B.begin(), B.end());

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

if(visited[i] || B[i].second==i)

continue;

int cycleSize=0;

int j=i;

while(!visited[j]){

visited[j]=true;

j=B[j].second;

cycleSize++;

}

if(cycleSize > 0)

res+= (cycleSize -1);

}

return res;

}

## [Strongly Connected Components](https://practice.geeksforgeeks.org/problems/strongly-connected-components-kosarajus-algo/1)

## [Shortest Source to Destination Path](https://practice.geeksforgeeks.org/problems/shortest-source-to-destination-path/0)

Company Tags [Samsung](https://practice.geeksforgeeks.org/company/Samsung/)

[**Problems**](https://practice.geeksforgeeks.org/problems/shortest-source-to-destination-path/0#problems)

Given a boolean 2D matrix (0-based index), find whether there is path from (0,0) to (x,y) and if there is one path, print the minimum no of steps needed to reach it, else print -1 if the destination is not reachable. You may move in only four direction ie up, down, left and right. The path can only be created out of a cell if its value is 1.  
  
**Input:**  
The first line of input contains an integer T denoting the no of test cases. Then T test cases follow. Each test case contains two lines . The first line of each test case contains two integers n and m denoting the size of the matrix. Then in the next line are n\*m space separated values of the matrix. The following line after it contains two integers x and y denoting the index of the destination.  
  
**Output:**  
For each test case print in a new line the min no of steps needed to reach the destination.  
  
**Constraints:**  
1<=T<=100  
1<=n,m<=20  
  
**Example:  
Input:**  
2  
3 4  
1 0 0 0 1 1 0 1 0 1 1 1  
2 3  
3 4  
1 1 1 1 0 0 0 1 0 0 0 1  
0 3  
**Output:**  
5  
3

The idea is inspired from [Lee algorithm](https://en.wikipedia.org/wiki/Lee_algorithm) and uses BFS.

1. We start from the source cell and calls BFS procedure.
2. We maintain a queue to store the coordinates of the matrix and initialize it with the source cell.
3. We also maintain a Boolean array visited of same size as our input matrix and initialize all its elements to false.
   1. We LOOP till queue is not empty
   2. Dequeue front cell from the queue
   3. Return if the destination coordinates have reached.
   4. For each of its four adjacent cells, if the value is 1 and they are not visited yet, we enqueue it in the queue and also mark them as visited.

#include <iostream>

#include <vector>

#include <queue>

using namespace std;

struct cell{

int x, y, dist;

cell(){};

cell(int X, int Y, int D):x(X), y(Y), dist(D){};

};

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

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

bool isSafe(int x, int y, int M, int N){

return (x>=0 && x<M && y>=0 && y<N);

}

int shortestDist(vector<vector<int>> &grid, int M, int N, int X, int Y){

if(grid[0][0]!=1 || grid[X][Y]!=1) return -1;

vector<vector<int>> visited(M, vector<int>(N, false));

queue<cell> Q;

Q.push(cell(0,0,0));

while(!Q.empty()){

cell c=Q.front();

Q.pop();

if(c.x==X && c.y==Y) return c.dist;

visited[c.x][c.y]=true;

for(int i=0; i<4; i++){

int x=c.x+dx[i];

int y=c.y+dy[i];

if(!isSafe(x,y,M,N)) continue;

if(grid[x][y]!=1) continue;

if(visited[x][y]) continue;

Q.push(cell(x,y,c.dist+1));

}

}

return -1;

}

int main() {

//code

int t,M,N,X,Y;

vector<vector<int>>grid;

cin>>t;

while(t--){

cin>>M>>N;

grid.resize(M, vector<int>(N));

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

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

cin>>grid[i][j];

}

cin>>X>>Y;

cout<<shortestDist(grid,M,N,X,Y)<<endl;

grid.clear();

}

return 0;

}

## [Find whether path exist](https://practice.geeksforgeeks.org/problems/find-whether-path-exist/0)

**Find whether path exist**

Company Tags [Adobe](https://practice.geeksforgeeks.org/company/Adobe/)  [Amazon](https://practice.geeksforgeeks.org/company/Amazon/)  [Morgan Stanley](https://practice.geeksforgeeks.org/company/Morgan%20Stanley/)  [Samsung](https://practice.geeksforgeeks.org/company/Samsung/)

* [**Problems**](https://practice.geeksforgeeks.org/problems/find-whether-path-exist/0#problems)

Given a **N** X **N** matrix (**M**) filled with 1, 0, 2, 3. The task is to find whether there is a path possible from source to destination, while traversing through blank cells only. You can traverse up, down, right and left.

* A value of cell **1** means Source.
* A value of cell **2** means Destination.
* A value of cell **3** means Blank cell.
* A value of cell **0**means Blank Wall.

**Note**: there is only single source and single destination.

**Input:**  
The first line of input is an integer **T** denoting the no of testcases. Then T test cases follow. Each test case consists of 2 lines. The first line of each test case contains an integer N denoting the size of the square matrix. Then in the next line are N\*N space separated values of the matrix (M).

**Output:**  
For each test case in a new line print 1 if the path exist from source to destination else print 0.

**Constraints:**  
1 <= T <= 20  
1 <= N <= 20

**Example:  
Input:**  
2  
4  
3 0 0 0 0 3 3 0 0 1 0 3 0 2 3 3   
3  
0 3 2 3 0 0 1 0 0

**Output:**  
1  
0

**Explanation:  
Testcase 1:**The matrix for the above given input is:  
3 0 0 0  
0 3 3 0  
0 1 0 3  
0 2 3 3  
From the matrix we can see that there exists a path from to reach destination 2 from source 1.  
**Testcase 2:** The matrix for the above given input is:  
0 3 2  
3 0 0  
1 0 0  
From the matrix we can see that there does not exists any path to reach destination 2 from source 1.

#include <iostream>

#include <vector>

#include <queue>

using namespace std;

struct cell{

int x, y;

cell(int X, int Y): x(X), y(Y){}

};

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

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

bool isSafe(int x, int y, int N){

return (x>=0 && x<N && y>=0 && y<N);

}

cell findSource(vector<vector<int>> &grid, int N){

//cell res(-1,-1);

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

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

if(grid[i][j]==1){

//cout<<"src found"<<endl;

return cell(i,j);

}

}

}

return cell(-1,-1);

}

bool isPathExist(vector<vector<int>> &grid, int N){

vector<vector<bool>> visited(N,vector<bool>(N,false));

cell src=findSource(grid,N);

//cout<<src.x<<" "<<src.y<<endl;

if(src.x==-1 || src.y==-1) return false;

queue<cell>Q;

Q.push(src);

while(!Q.empty()){

cell c=Q.front();

Q.pop();

if(grid[c.x][c.y]==2) return true;

visited[c.x][c.y]=true;

for(int i=0; i<4; i++){

int x=c.x+dx[i];

int y=c.y+dy[i];

if(!isSafe(x,y,N)) continue;

if(visited[x][y]) continue;

if(grid[x][y]==0) continue;

Q.push(cell(x,y));

}

}

return false;

}

int main() {

//code

int t, N;

vector<vector<int>>grid;

cin>>t;

while(t--){

cin>>N;

grid.resize(N, vector<int>(N));

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

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

cin>>grid[i][j];

cout<<isPathExist(grid, N)<<endl;

grid.clear();

}

return 0;

}

## [Minimum Cost Path](https://practice.geeksforgeeks.org/problems/minimum-cost-path/0)

**Minimum Cost Path**

Company Tags [Goldman Sachs](https://practice.geeksforgeeks.org/company/Goldman%20Sachs/)  [MakeMyTrip](https://practice.geeksforgeeks.org/company/MakeMyTrip/)  [Samsung](https://practice.geeksforgeeks.org/company/Samsung/)

* [**Problems**](https://practice.geeksforgeeks.org/problems/minimum-cost-path/0#problems)

Given a square grid of size **N**, each cell of which contains integer cost which represents a cost to traverse through that cell, we need to find a path from top left cell to bottom right cell by which total cost incurred is minimum.

**Note :**It is assumed that negative cost cycles do not exist in input matrix.

**Input:**  
The first line of input will contain number of testcases **T**. Then T test cases follow. Each test case contains 2 lines. The first line of each test case contains an integer N denoting the size of the grid. Next line of each test contains a single line containing N\*N space separated integers depicting the cost of respective cell from (0,0) to (N,N).

**Output:**  
For each test case output a single integer depecting the minimum cost to reach the destination.

**Constraints:**  
1 <= T <= 50  
1 <= N <= 50

**Example:  
Input:**  
2  
5  
31 100 65 12 18 10 13 47 157 6 100 113 174 11 33 88 124 41 20 140 99 32 111 41 20  
2  
42 93 7 14

**Output**:  
327  
63

**Explanation:**  
**Testcase 1:**  
Grid is:  
31, 100, 65, 12, 18,  
10, 13, 47, 157, 6,  
100. 113, 174, 11, 33,  
88, 124, 41, 20, 140,  
99, 32, 111, 41, 20  
A cost grid is given in below diagram, minimum  
cost to reach bottom right from top left  
is 327 (31 + 10 + 13 + 47 + 65 + 12 + 18 + 6 + 33 + 11 + 20 + 41 + 20)

Sol:

#include <iostream>

#include <vector>

#include <set>

#include <climits>

using namespace std;

struct cell{

int x,y, dist;

cell(int X, int Y, int D): x(X), y(Y), dist(D){}

};

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

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

bool operator <(const cell &a, const cell &b){

if(a.dist==b.dist){

if(a.x!=b.x) return a.x<b.x;

else return a.y<b.y;

}

return a.dist<b.dist;

}

bool isSafe(int x, int y, int N){

return (x>=0 && x<N && y>=0 && y<N);

}

int minCost(vector<vector<int>> &grid){

int N=grid.size();

vector<vector<int>>dp(N,vector<int>(N, INT\_MAX));

dp[0][0]=grid[0][0];

set<cell>S;

S.insert(cell(0,0,0));

while(!S.empty()){

cell c=\*S.begin();

S.erase(S.begin());

for(int i=0; i<4; i++){

int x=c.x+dx[i];

int y=c.y+dy[i];

if(!isSafe(x,y,N)) continue;

if(dp[x][y]>dp[c.x][c.y]+grid[x][y]){

if(dp[x][y]!=INT\_MAX)

S.erase(S.find(cell(x,y,dp[x][y])));

dp[x][y]=dp[c.x][c.y]+grid[x][y];

S.insert(cell(x,y,dp[x][y]));

}

}

}

return dp[N-1][N-1];

}

int main() {

//code

int t,N;

vector<vector<int>>grid;

cin>>t;

while(t--){

cin>>N;

grid.resize(N,vector<int>(N));

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

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

cin>>grid[i][j];

cout<<minCost(grid)<<endl;

grid.clear();

}

return 0;

}

## [Circle of Strings](https://practice.geeksforgeeks.org/problems/circle-of-strings/0)

    C++ code to check if cyclic order is possible among strings

//  under given constrainsts

#include <bits/stdc++.h>

using namespace std;

#define M 26

//    Utility method for a depth first search among vertices

void dfs(vector<int> g[], int u, vector<bool> &visit)

{

    visit[u] = true;

    for (int i = 0; i < g[u].size(); ++i)

        if(!visit[g[u][i]])

            dfs(g, g[u][i], visit);

}

//    Returns true if all vertices are strongly connected

// i.e. can be made as loop

bool isConnected(vector<int> g[], vector<bool> &mark, int s)

{

    // Initialize all vertices as not visited

    vector<bool> visit(M, false);

    //    perform a dfs from s

    dfs(g, s, visit);

    //    now loop through all characters

    for (int i = 0; i < M; i++)

    {

        /\*  I character is marked (i.e. it was first or last

            character of some string) then it should be

            visited in last dfs (as for looping, graph

            should be strongly connected) \*/

        if (mark[i] && !visit[i])

            return false;

    }

    //    If we reach that means graph is connected

    return true;

}

//    return true if an order among strings is possible

bool possibleOrderAmongString(string arr[], int N)

{

    // Create an empty graph

    vector<int> g[M];

    // Initialize all vertices as not marked

    vector<bool> mark(M, false);

    // Initialize indegree and outdegree of every

    // vertex as 0.

    vector<int> in(M, 0), out(M, 0);

    // Process all strings one by one

    for (int i = 0; i < N; i++)

    {

        // Find first and last characters

        int f = arr[i].front() - 'a';

        int l = arr[i].back() - 'a';

        // Mark the characters

        mark[f] = mark[l] = true;

        //    increase indegree and outdegree count

        in[l]++;

        out[f]++;

        // Add an edge in graph

        g[f].push\_back(l);

    }

    // If for any character indegree is not equal to

    // outdegree then ordering is not possible

    for (int i = 0; i < M; i++)

        if (in[i] != out[i])

            return false;

    return isConnected(g, mark, arr[0].front() - 'a');

}

## [Floyd Warshall](https://practice.geeksforgeeks.org/problems/implementing-floyd-warshall/0)

Associated Course(s): [Must Do Interview Preparation](https://practice.geeksforgeeks.org/courses/must-do-interview-prep/)

Show Topic Tags

Company Tags [Samsung](https://practice.geeksforgeeks.org/company/Samsung/)

* [**Problems**](https://practice.geeksforgeeks.org/problems/implementing-floyd-warshall/0#problems)

The problem is to find shortest distances between every pair of vertices in a given edge weighted directed Graph. The Graph is represented as Adjancency Matrix, and the Matrix denotes the weight of the edegs (if it exists) else INF (1e7).  
  
**Input:**  
The first line of input contains an integer **T** denoting the no of test cases. Then T test cases follow. The first line of each test case contains an integer V denoting the size of the adjacency matrix. The next V lines contain V space separated values of the matrix (graph). All input will be integer type.

**Output:**  
For each test case output will be V\*V space separated integers where the i-jth integer denote the shortest distance of ith vertex from jth vertex. For INT\_MAX integers output INF.  
  
**Constraints:**  
1 <= T <= 20   
1 <= V <= 100  
1 <= graph[][] <= 500  
  
**Example:  
Input**  
2  
2  
0 25  
INF 0  
3  
0 1 43  
1 0 6  
INF INF 0

**Output**  
0 25  
INF 0   
0 1 7  
1 0 6  
INF INF 0

#include <iostream>

#include <vector>

using namespace std;

#define INF 10000000

void FloydWarshall(vector<vector<int>> &graph){

int n=graph.size();

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

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

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

if(graph[i][k]>=INF || graph[k][j]>=INF) continue;

if(graph[i][j]>graph[i][k]+graph[k][j])

graph[i][j]=graph[i][k]+graph[k][j];

}

}

}

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

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

if(graph[i][j]>=INF)

cout<<"INF"<<" ";

else

cout<<to\_string(graph[i][j])<<" ";

}

cout<<endl;

}

}

int main() {

//code

int t,v;

vector<vector<int>> graph;

cin>>t;

while(t--){

cin>>v;

graph.resize(v, vector<int>(v));

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

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

cin>>graph[i][j];

}

FloydWarshall(graph);

graph.clear();

}

return 0;

}

## [Alien Dictionary](https://practice.geeksforgeeks.org/problems/alien-dictionary/1)

Find order of alphabets

Company Tags [Amazon](https://practice.geeksforgeeks.org/company/Amazon/)  [Google](https://practice.geeksforgeeks.org/company/Google/)  [Microsoft](https://practice.geeksforgeeks.org/company/Microsoft/)  [OYO Rooms](https://practice.geeksforgeeks.org/company/OYO%20Rooms/)  [Walmart](https://practice.geeksforgeeks.org/company/Walmart/)

* [**Problems**](https://practice.geeksforgeeks.org/problems/alien-dictionary/1#problems)

Given a sorted dictionary of an alien language having N words and k starting alphabets of standard dictionary the task is to complete the function which returns a string denoting the order of characters in the language.  
**Note:** Many orders may be possible for a particular test case, thus you may return any valid order.  
**Examples:**  
Input:  Dict[] = { "baa", "abcd", "abca", "cab", "cad" }, k = 4  
Output: Function returns "bdac"  
Here order of characters is 'b', 'd', 'a', 'c'  
Note that words are sorted and in the given language "baa"  
comes before "abcd", therefore 'b' is before 'a' in output.  
Similarly we can find other orders.

Input: Dict[] = { "caa", "aaa", "aab" }, k = 3  
Output: Function returns "cab"

**Input:**  
The first line of input contains an integer T denoting the no of test cases. Then T test cases follow. Each test case contains an integer N and k denoting the size of the dictionary. Then in the next line are sorted space separated values of the alien dictionary.

**Output:**  
For each test case in a new line output will be 1 if the order of string returned by the function is correct else 0 denoting incorrect string returned.

**Constraints:**  
1 <= T <= 200  
1 <= N <= 100  
1 <= k <= 26  
1 <= Length of words <= 1000

**Example:  
Input:**  
2  
5 4  
baa abcd abca cab cad  
3 3  
caa aaa aab

**Output:**  
1  
1

|  |
| --- |
| // A C++ program to order of characters in an alien language  #include<iostream>  #include <list>  #include <stack>  #include <cstring>  using namespace std;    // Class to represent a graph  class Graph  {      int V;    // No. of vertices'        // Pointer to an array containing adjacency listsList      list<int> \*adj;        // A function used by topologicalSort      void topologicalSortUtil(int v, bool visited[], stack<int> &Stack);  public:      Graph(int V);   // Constructor        // function to add an edge to graph      void addEdge(int v, int w);        // prints a Topological Sort of the complete graph      void topologicalSort();  };    Graph::Graph(int V)  {      this->V = V;      adj = new list<int>[V];  }    void Graph::addEdge(int v, int w)  {      adj[v].push\_back(w); // Add w to v’s list.  }    // A recursive function used by topologicalSort  void Graph::topologicalSortUtil(int v, bool visited[], stack<int> &Stack)  {      // Mark the current node as visited.      visited[v] = true;        // Recur for all the vertices adjacent to this vertex      list<int>::iterator i;      for (i = adj[v].begin(); i != adj[v].end(); ++i)          if (!visited[\*i])              topologicalSortUtil(\*i, visited, Stack);        // Push current vertex to stack which stores result      Stack.push(v);  }    // The function to do Topological Sort. It uses recursive topologicalSortUtil()  void Graph::topologicalSort()  {      stack<int> Stack;        // Mark all the vertices as not visited      bool \*visited = new bool[V];      for (int i = 0; i < V; i++)          visited[i] = false;        // Call the recursive helper function to store Topological Sort      // starting from all vertices one by one      for (int i = 0; i < V; i++)          if (visited[i] == false)              topologicalSortUtil(i, visited, Stack);        // Print contents of stack      while (Stack.empty() == false)      {          cout << (char) ('a' + Stack.top()) << " ";          Stack.pop();      }  }    int min(int x, int y)  {      return (x < y)? x : y;  }    // This function fidns and prints order of characer from a sorted  // array of words. n is size of words[].  alpha is set of possible  // alphabets.  // For simplicity, this function is written in a way that only  // first 'alpha' characters can be there in words array.  For  // example if alpha is 7, then words[] should have only 'a', 'b',  // 'c' 'd', 'e', 'f', 'g'  void printOrder(string words[], int n, int alpha)  {      // Create a graph with 'alpha' edges      Graph g(alpha);        // Process all adjacent pairs of words and create a graph      for (int i = 0; i < n-1; i++)      {          // Take the current two words and find the first mismatching          // character          string word1 = words[i], word2 = words[i+1];          for (int j = 0; j < min(word1.length(), word2.length()); j++)          {              // If we find a mismatching character, then add an edge              // from character of word1 to that of word2              if (word1[j] != word2[j])              {                  g.addEdge(word1[j]-'a', word2[j]-'a');                  break;              }          }      }        // Print topological sort of the above created graph      g.topologicalSort();  } |

**953. Verifying an Alien Dictionary (LeetCode)**

Easy

297107FavoriteShare

In an alien language, surprisingly they also use english lowercase letters, but possibly in a different order. The order of the alphabet is some permutation of lowercase letters.

Given a sequence of words written in the alien language, and the order of the alphabet, return true if and only if the given words are sorted lexicographicaly in this alien language.

**Example 1:**

**Input:** words = ["hello","leetcode"], order = "hlabcdefgijkmnopqrstuvwxyz"

**Output:** true

**Explanation:** As 'h' comes before 'l' in this language, then the sequence is sorted.

**Example 2:**

**Input:** words = ["word","world","row"], order = "worldabcefghijkmnpqstuvxyz"

**Output:** false

**Explanation:** As 'd' comes after 'l' in this language, then words[0] > words[1], hence the sequence is unsorted.

**Example 3:**

**Input:** words = ["apple","app"], order = "abcdefghijklmnopqrstuvwxyz"

**Output:** false

**Explanation:** The first three characters "app" match, and the second string is shorter (in size.) According to lexicographical rules "apple" > "app", because 'l' > '∅', where '∅' is defined as the blank character which is less than any other character ([More info](https://en.wikipedia.org/wiki/Lexicographical_order)).

**Note:**

1. 1 <= words.length <= 100
2. 1 <= words[i].length <= 20
3. order.length == 26
4. All characters in words[i] and order are english lowercase letters.

class Solution {

public:

bool isAlienSorted(vector<string>& words, string order) {

for (int i = 0; i < order.size(); ++i)

od[order[i]] = i;

for (int i = 0 ; i < words.size() - 1; ++i)

{

if (Greater(words[i] , words[i+1]))

return false;

}

return true;

}

bool Greater(string A, string B)

{

int n = min(A.size(), B.size());

int i = 0;

while(i < n)

{

if ( od[A[i]] > od[B[i]])

return true;

else if (od[A[i]] < od[B[i]])

return false;

else

++i;

}

return A.size() > B.size();

}

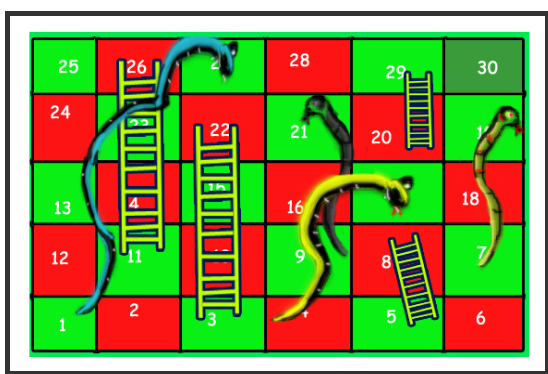
private:

unordered\_map<char, int> od;

};

## [Snake and Ladder Problem](https://practice.geeksforgeeks.org/problems/snake-and-ladder-problem/0)

[Adobe](https://practice.geeksforgeeks.org/company/Adobe/)  [Amazon](https://practice.geeksforgeeks.org/company/Amazon/)  [Belzabar](https://practice.geeksforgeeks.org/company/Belzabar/)  [Flipkart](https://practice.geeksforgeeks.org/company/Flipkart/)  [MAQ Software](https://practice.geeksforgeeks.org/company/MAQ%20Software/)  [Nutanix](https://practice.geeksforgeeks.org/company/Nutanix/)  [OYO Rooms](https://practice.geeksforgeeks.org/company/OYO%20Rooms/)  [Zoho](https://practice.geeksforgeeks.org/company/Zoho/)

Given a snake and ladder board of order 5x6, find the minimum number of dice throws required to reach the destination or last cell (30th cell) from source (1st cell) .   
  
Example

​For the above board output will be 3   
For 1st throw get a 2  
For 2nd throw get a 6  
For 3rd throw get a 2

**Input:**  
The first line of input contains an integer T denoting the no of test cases. Then T test cases follow. Each test case contains two lines. The first line of input contains an integer N denoting the no of ladders and snakes present. Then in the next line are 2\*N space separated values a,b which denotes a ladder or a snake at position **'a'** which takes to a position **'b'**.

**Output:**  
For each test case in a new line print the required answer denoting the min no of dice throws.

**Constraints:**  
1 <= T <= 100  
1 <= N <= 10  
1 <= a <= 30  
1 <= b <= 30

**Example:  
Input:**  
2  
6  
11 26 3 22 5 8 20 29 27 1 21 9  
1  
2 30

**Output:**  
3  
1

**Explanation:  
Testcase 1:**  
For 1st throw get a 2, which contains ladder to reach 22  
For 2nd throw get a 6, which will lead to 28  
Finally get a 2, to reach at the end 30. Thus 3 dice throws required to reach 30.

#### Approach: The game grid is numbered from 1 to N. So create vector of N, store ladder n snake values in that. Solve using BFS.

#include<iostream>

#include<vector>

#include<queue>

using namespace std;

struct qNode{

int vertex, throws;

qNode(){};

qNode(int v, int t): vertex(v), throws(t){};

};

int solve(int N, vector<int>&board){

vector<bool> visited(N,false);

queue<qNode> Q;

visited[0]=true;

Q.push(qNode(0,0)); //start cell

qNode qfront;

while(!Q.empty()){

qfront=Q.front();

//cout<<qfront.vertex<<" "<<qfront.throws<<endl;

Q.pop();

int v=qfront.vertex;

if(v==N-1) break;

for(int j=v+1; j<=(v+6) && j<N; j++){

if(!visited[j]){

visited[j]=true;

qNode child;

child.throws=qfront.throws+1;

if(board[j]!=-1)

child.vertex=board[j];

else

child.vertex=j;

Q.push(child);

}

}

}

return qfront.throws;

}

int main() {

//code

int N=30;// total cells on board

int t, m;

vector<int> board(N,-1);

cin>>t;

while(t--){

board.resize(N,-1);

cin>>m;

int s, d;

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

{

cin>>s>>d;

board[s-1]=d-1;

}

cout<<solve(N, board)<<endl;

board.clear();

}

return 0;

}

# **Greedy**

## [Activity Selection](https://practice.geeksforgeeks.org/problems/activity-selection/0)

## [N meetings in one room](https://practice.geeksforgeeks.org/problems/n-meetings-in-one-room/0)

## [Coin Piles](https://practice.geeksforgeeks.org/problems/coin-piles/0)

## [Maximize Toys](https://practice.geeksforgeeks.org/problems/maximize-toys/0)

## [Page Faults in LRU](https://practice.geeksforgeeks.org/problems/page-faults-in-lru/0)

## [Largest number possible](https://practice.geeksforgeeks.org/problems/largest-number-possible/0)

## [Minimize the heights](https://practice.geeksforgeeks.org/problems/minimize-the-heights/0)

## [Minimize the sum of product](https://practice.geeksforgeeks.org/problems/minimize-the-sum-of-product/0)

## [Huffman Decoding](https://practice.geeksforgeeks.org/problems/huffman-decoding-1/1)

## [Minimum Spanning Tree](https://practice.geeksforgeeks.org/problems/minimum-spanning-tree/1)

Company Tags [Amazon](https://practice.geeksforgeeks.org/company/Amazon/)  [Cisco](https://practice.geeksforgeeks.org/company/Cisco/)  [Samsung](https://practice.geeksforgeeks.org/company/Samsung/)

* [**Problems**](https://practice.geeksforgeeks.org/problems/minimum-spanning-tree/1#problems)

Given a weighted, undirected and connected graph. The task is to find the sum of weights of  the edges of the Minimum Spanning Tree.

**Input:**  
The first line of input contains an integer **T** denoting the number of testcases. Then T test cases follow. The first line of each testcase contains two integers N (starting from 1), E denoting the number of nodes and number of edges. Then in the next line are 3\*E space separated values a b w where a, b denotes an **edge** from **a** to **b** and **w** is the weight of the edge.

**Output:**  
For each test case in a new line print the sum of weights of  the edges of the Minimum Spanning Tree formed of the graph.

**User task:**  
Since this is a functional problem you don't have to worry about input, you just have to complete the function  **spanningTree()** which takes a graph **g** as its argument and returns an integer denoting the sum of weights of the edges of the Minimum Spanning Tree.

**Constraints:**  
1 <= T <= 10  
1 <= N<= 100  
N-1 <= E<= 1000  
1 <= w <= 1000

**Example:  
Input**:  
2  
3 3  
1 2 5 2 3 3 1 3 1  
2 1  
1 2 5

**Output**:  
4  
5

#### Sol: DisjointSet create, find, merge

struct myclass{

bool operator()(const vector<int> &A, const vector<int> &B){

return A[2]<B[2];

}

}myobj;

class DisjointSet{

private:

int noC;

vector<int> parent, rank;

public:

DisjointSet(int n): noC(n), parent(n+1,-1), rank(n+1,1){

resetParent();

}

void resetParent(){

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

parent[i]=i;

}

int getRootParent(int a){

int p=parent[a];

while(p!=parent[p])

p=parent[p];

return p;

}

bool connected(int a, int b){

int rootA=getRootParent(a);

int rootB=getRootParent(b);

return (rootA==rootB);

}

int getNoC(){

return noC;

}

void merge(int a, int b){

int rootA=getRootParent(a);

int rootB=getRootParent(b);

if(rootA==rootB) return;

if(rank[rootA]>rank[rootB]){

parent[rootB]=rootA;

rank[rootA]+=rank[rootB];

}

else{

parent[rootA]=rootB;

rank[rootB]+=rank[rootA];

}

noC--;

}

};

/\*This is a function problem.You only need to complete the function given below\*/

// Function to construct and print MST for

// a graph represented using adjacency

// matrix representation, with V vertices.

// graph[i][j] = weight if edge exits else INT\_MAX

int spanningTree(int N,int E,vector<vector<int> > graph)

{

// code here

vector<vector<int>> edges;

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

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

if(i!=j && graph[i][j]!=INT\_MAX)

edges.push\_back(vector<int>{i+1,j+1,graph[i][j]});

}

}

DisjointSet DS(N);

sort(edges.begin(), edges.end(), myobj);

int sum=0;

int i=0;

while(DS.getNoC()!=1){

vector<int> e=edges[i];

if(!DS.connected(e[0], e[1])){

DS.merge(e[0], e[1]);

sum+=e[2];

}

i++;

}

return sum;

}

## [Shop in Candy Store](https://practice.geeksforgeeks.org/problems/shop-in-candy-store/0)

* [**Problems**](https://practice.geeksforgeeks.org/problems/shop-in-candy-store/0#problems)

In a candy store there are **N** different types of candies available  and the prices of all the N different types of candies are provided to you.  
You are now provided with an attractive offer.  
You can buy a single candy from the store and get atmost **K** other candies ( all are different types ) for free.  
Now you have to answer two questions. Firstly, you have to tell what is the **minimum amount of money** you have to spend to buy all the**N**different candies. Secondly, you have to tell what is the **maximum amount of money** you have to spend to buy all the N different candies.  
In both the cases you must utilize the offer i.e. you buy one candy and get **K**other candies for free.

**Input**   
The first line of the input contains **T** the number of test cases. Each test case consists of two lines. The first line of each test case contains the values of **N** and **K** as described above.  Then in the next line **N** integers follow denoting the price of each of the**N** different candies.

**Output**  
For each test case output a single line containing **2** space separated integers , the first denoting the **minimum amount of money required to be spent**and the second denoting the **maximum amount of money to be spent**.  
Remember to output the answer of each test case in a new line.  
  
**Constraints**        
1 <= **T**<= 50  
1 <= **N**<= 1000  
 0 <= **K** <= N-1  
1 <= **Ai** <= 100

**Expected Time Complexity :**O(nlogn)

**Example:**  
**Input**     
 1  
 4  2  
 3 2 1 4

**Output**  
3 7

**Explanation**  
As according to the offer if you but one candy you can take atmost two more for free.  
So in the first case you buy the candy which costs 1 and take candies worth 3 and 4 for free, also you buy candy worth 2 as well.  
So **min cost** = 1+2 =3.  
In the second case I buy the candy which costs 4 and take candies worth 1 and 2 for free, also I  buy candy worth 3 as well.  
So **max cost** = 3+4 =7.

#include <iostream>

#include<bits/stdc++.h>

using namespace std;

int findMin(vector<int> &A, int n, int k){

int res=0;

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

res+=A[i];

n=n-k;

}

return res;

}

int findMax(vector<int> &A, int n, int k){

int res=0, start=0;

for(int i=n-1; i>=start; i--){

res+=A[i];

start+=k;

}

return res;

}

int main() {

//code

int t,n,k;

vector<int>A;

cin>>t;

while(t--){

cin>>n>>k;

A.resize(n);

for(int i=0; i<n; i++) cin>>A[i];

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

cout<<findMin(A,n,k)<<" "<<findMax(A,n,k)<<endl;

}

return 0;

}

## [Geek collects the balls](https://practice.geeksforgeeks.org/problems/geek-collects-the-balls/0)

# **Dynamic Programming**

## [Minimum Operations](https://practice.geeksforgeeks.org/problems/find-optimum-operation/0)

* [**Problems**](https://practice.geeksforgeeks.org/problems/find-optimum-operation/0#problems)

You are given a number**N**.You have to find the number of operations required to reach **N** from **0**. You have 2 operations available:

* Double the number
* Add one to the number

**Input:**  
The first line of input contains an integer T denoting the number of test cases. Then T test cases follow. Each test case contains an integer **N**.

**Output:**  
For each test case, in a new line, print the **minimum**number of operations required to reach **N from 0**.

**Constraints:**  
1<=T<=100  
1<=N<=104

**Example:  
Input:**  
2  
8  
7  
**Input:**  
4  
5

**Explanation:**  
**Testcase1:**  
**Input  :** N = 8  
**Output :** 4  
0 + 1 = 1, 1 + 1 = 2, 2 \* 2 = 4, 4 \* 2 = 8  
**Testcase2:**  
**Input  :** N = 7  
**Output :** 5  
0 + 1 = 1, 1 + 1 = 2, 1 + 2 = 3, 3 \* 2 = 6, 6 + 1 = 7

#include <iostream>

#include<bits/stdc++.h>

using namespace std;

int steps(int n){

vector<int> dp(n+1,0);

dp[0]=0;

dp[1]=1;

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

if(i%2)

dp[i]=dp[i-1]+1;

else

dp[i]=min(dp[i-1],dp[i/2])+1;

}

return dp[n];

}

int main() {

//code

int t,n;

cin>>t;

while(t--){

cin>>n;

cout<<steps(n)<<endl;

}

return 0;

}

## [Max length chain](https://practice.geeksforgeeks.org/problems/max-length-chain/1)

Company Tags [Amazon](https://practice.geeksforgeeks.org/company/Amazon/)

* [**Problems**](https://practice.geeksforgeeks.org/problems/max-length-chain/1#problems)

You are given N pairs of numbers. In every pair, the first number is always smaller than the second number. A pair (c, d) can follow another pair (a, b) if b < c. Chain of pairs can be formed in this fashion. Your task is to complete the function **maxChainLen** which returns an integer denoting the longest chain which can be formed from a given set of pairs.   
  
**Input:**  
The first line of input contains an integer T denoting the no of test cases then T test cases follow .Then T test cases follow . The first line of input contains an integer N denoting the no of pairs . In the next line are 2\*N space separated values denoting N pairs.  
  
**Output:**  
For each test case output will be the length of the longest chain formed.  
  
**Constraints:**  
1<=T<=100  
1<=N<=100  
  
**Example(To be used only for expected output):  
Input**  
2  
5  
5  24 39 60 15 28 27 40 50 90  
2  
5 10 1 11   
  
**Output**  
3  
1  
​  
**Explanation**  
(i) the given pairs are {{5, 24}, {39, 60}, {15, 28}, {27, 40}, {50, 90} },the longest chain that can be formed is of length 3, and the chain is {{5, 24}, {27, 40}, {50, 90}}  
(ii) The max length chain possible is only of length one.

#### Sol:

struct myclass{

bool operator()(const vector<int>&A, const vector<int>&B){

return (A[0]<B[0]);

}

}myobj;

class Solution {

public:

int findLongestChain(vector<vector<int>>& pairs) {

int n=pairs.size();

if(n<2) return n;

sort(pairs.begin(), pairs.end());

vector<int>lis(n, 1);

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

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

if(pairs[i][0]>pairs[j][1] && lis[i]<lis[j]+1)

lis[i]=lis[j]+1;

}

return \*max\_element(lis.begin(),lis.end());

}

};

## [Minimum number of Coins](https://practice.geeksforgeeks.org/problems/-minimum-number-of-coins/0)

Company Tags [Visa](https://practice.geeksforgeeks.org/company/Visa/)

* [**Problems**](https://practice.geeksforgeeks.org/problems/-minimum-number-of-coins/0#problems)

Given a value **N**, total sum you have. You have to make change for Rs. N, and there is infinite supply of each of the denominations in Indian currency, i.e., you have infinite supply of { 1, 2, 5, 10, 20, 50, 100, 200, 500, 2000} valued coins/notes, Find the minimum number of coins and/or notes needed to make the change for Rs N.

**Input:**  
The first line of input contains an integer T denoting the number of test cases. Each test case consist of an Integer value N denoting the amount to get change for.

**Output:**  
Print all the denominations needed to make the change in a separate line.

**Constraints:**  
1 ≤ T ≤ 100  
1 ≤ N ≤ 106

**Example:**  
**Input:**  
1  
43

**Output:**  
20 20 2 1

**Explanation:  
Testcase 1:** Sum of Rs 43 can be changed with minimum of 4 coins/ notes 20, 20, 2, 1.

#### Sol:

#include <iostream>

#include<bits/stdc++.h>

using namespace std;

void split(int n){

while(n>0){

if(n>=2000){

n-=2000;

cout<<2000<<" ";

}

else if(n>=500){

n-=500;

cout<<500<<" ";

}

else if(n>=200){

n-=200;

cout<<200<<" ";

}

else if(n>=100){

n-=100;

cout<<100<<" ";

}

else if(n>=50){

n-=50;

cout<<50<<" ";

}

else if(n>=20){

n-=20;

cout<<20<<" ";

}

else if(n>=10){

n-=10;

cout<<10<<" ";

}

else if(n>=5){

n-=5;

cout<<5<<" ";

}

else if(n>=2){

n-=2;

cout<<2<<" ";

}

else if(n>=1){

n-=1;

cout<<1<<" ";

}

}

}

int main() {

//code

int t,n;

cin>>t;

while(t--){

cin>>n;

split(n);

cout<<endl;

}

return 0;

}

## [Longest Common Substring](https://practice.geeksforgeeks.org/problems/longest-common-substring/0)

## [Longest Increasing Subsequence](https://practice.geeksforgeeks.org/problems/longest-increasing-subsequence/0)

## [Longest Common Subsequence](https://practice.geeksforgeeks.org/problems/longest-common-subsequence/0)

## [0 – 1 Knapsack Problem](https://practice.geeksforgeeks.org/problems/0-1-knapsack-problem/0)

## [Maximum sum increasing subsequence](https://practice.geeksforgeeks.org/problems/maximum-sum-increasing-subsequence/0)

## [Minimum number of jumps](https://practice.geeksforgeeks.org/problems/minimum-number-of-jumps/0)

## [Edit Distance](https://practice.geeksforgeeks.org/problems/edit-distance/0)

## [Coin Change Problem](https://practice.geeksforgeeks.org/problems/coin-change/0)

Company Tags [Accolite](https://practice.geeksforgeeks.org/company/Accolite/)  [Amazon](https://practice.geeksforgeeks.org/company/Amazon/)  [Microsoft](https://practice.geeksforgeeks.org/company/Microsoft/)  [Morgan Stanley](https://practice.geeksforgeeks.org/company/Morgan%20Stanley/)  [OYO Rooms](https://practice.geeksforgeeks.org/company/OYO%20Rooms/)  [Paytm](https://practice.geeksforgeeks.org/company/Paytm/)  [Samsung](https://practice.geeksforgeeks.org/company/Samsung/)  [Snapdeal](https://practice.geeksforgeeks.org/company/Snapdeal/)

* [**Problems**](https://practice.geeksforgeeks.org/problems/coin-change/0#problems)

Given a value N, find the number of ways to make change for N cents, if we have infinite supply of each of S = { S1, S2, .. , Sm} valued coins. The order of coins doesn’t matter. For example, for N = 4 and S = {1,2,3}, there are four solutions: {1,1,1,1},{1,1,2},{2,2},{1,3}. So output should be 4. For N = 10 and S = {2, 5, 3, 6}, there are five solutions: {2,2,2,2,2}, {2,2,3,3}, {2,2,6}, {2,3,5} and {5,5}. So the output should be 5.

**Input:**  
The first line contains an integer '**T**' denoting the total number of test cases. In each test cases, the first line contains an integer '**M**' denoting the size of array. The second line contains M space-separated integers A1, A2, ..., AN denoting the elements of the array. The third line contains an integer 'N' denoting the cents.

**Output:**  
Print number of possible ways to make change for N cents.

**Constraints:**  
1 ≤ T ≤ 50  
1 ≤ N ≤ 300  
1 ≤ A[i] ≤ 300

**Example:  
Input:**  
2  
3  
1 2 3  
4  
4  
2 5 3 6  
10

**Output:**  
4  
5

#### Sol: count ways with and without A[i] is considered.

#include <iostream>

#include<bits/stdc++.h>

using namespace std;

int countWays(vector<int> &A, int sum, int n){

if(sum==0) return 1;

if(sum<0) return 0;

if(sum>0 && n==0) return 0;

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

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

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

int x=(A[j]<=i)?dp[i-A[j]][j]:0;

int y=(j>0)?dp[i][j-1]:0;

dp[i][j]=x+y;

}

}

return dp[sum][n-1];

}

int main() {

//code

int t;

int n, sum;

vector<int> A;

cin>>t;

while(t--){

cin>>n;

A.resize(n);

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

cin>>A[i];

cin>>sum;

cout<<countWays(A,sum,n)<<endl;

A.clear();

}

return 0;

}

## Subset Sum- Partition problems

## [Box Stacking](https://practice.geeksforgeeks.org/problems/box-stacking/1)

## [Rod Cutting](https://practice.geeksforgeeks.org/problems/cutted-segments/0)

## [Path in Matrix](https://www.geeksforgeeks.org/find-the-longest-path-in-a-matrix-with-given-constraints/)

## [Minimum sum partition](https://practice.geeksforgeeks.org/problems/minimum-sum-partition/0)

## [Count number of ways to cover a distance](https://practice.geeksforgeeks.org/problems/count-number-of-hops/0)

Company Tags [Amazon](https://practice.geeksforgeeks.org/company/Amazon/)

* [**Problems**](https://practice.geeksforgeeks.org/problems/count-number-of-hops/0#problems)

A frog jumps either **1**, **2** or **3** steps to go to top. In how many ways can it reach the top.

**Input:**  
The first line of input contains an integer T denoting the number of test cases. T testcases follow. Each testcase contains one line of input **N** denoting the total number of steps.

**Output:**  
For each testcase, in a new line, print the number of ways to reach the top.

**Constraints:**  
1 ≤ T ≤ 100  
1 ≤ N ≤ 50

**Example:  
Input:**  
2  
1  
5  
**Output:**  
1  
13

int steps(int n){

int dp[n+1];

dp[0]=0; dp[1]=1; dp[2]=2; dp[3]=4;

for(int i=4; i<=n; i++){

dp[i]=dp[i-1]+dp[i-2]+dp[i-3];

}

return dp[n];

}

## [Egg Dropping Puzzle](https://practice.geeksforgeeks.org/problems/egg-dropping-puzzle/0)

Company Tags [Amazon](https://practice.geeksforgeeks.org/company/Amazon/)  [D-E-Shaw](https://practice.geeksforgeeks.org/company/D-E-Shaw/)  [Goldman Sachs](https://practice.geeksforgeeks.org/company/Goldman%20Sachs/)  [Google](https://practice.geeksforgeeks.org/company/Google/)  [Hike](https://practice.geeksforgeeks.org/company/Hike/)  [MakeMyTrip](https://practice.geeksforgeeks.org/company/MakeMyTrip/)  [MAQ Software](https://practice.geeksforgeeks.org/company/MAQ%20Software/)  [Myntra](https://practice.geeksforgeeks.org/company/Myntra/)  [nearbuy](https://practice.geeksforgeeks.org/company/nearbuy/)  [Opera](https://practice.geeksforgeeks.org/company/Opera/)  [Oracle](https://practice.geeksforgeeks.org/company/Oracle/)  [Philips](https://practice.geeksforgeeks.org/company/Philips/)  [Samsung](https://practice.geeksforgeeks.org/company/Samsung/)  [Service Now](https://practice.geeksforgeeks.org/company/Service%20Now/)  [Unisys](https://practice.geeksforgeeks.org/company/Unisys/)  [VMWare](https://practice.geeksforgeeks.org/company/VMWare/)

* [**Problems**](https://practice.geeksforgeeks.org/problems/egg-dropping-puzzle/0#problems)

Suppose you have N eggs and you want to determine from which floor in a K-floor building you can drop an egg such that it doesn't break. You have to determine the minimum number of attempts you need in order find the critical floor in the worst case while using the best strategy.There are few rules given below.

* An egg that survives a fall can be used again.
* A broken egg must be discarded.
* The effect of a fall is the same for all eggs.
* If the egg doesn't break at a certain floor, it will not break at any floor below.
* If the eggs breaks at a certain floor, it will break at any floor above.

For more description on this problem see [wiki page](http://en.wikipedia.org/wiki/Dynamic_programming#Egg_dropping_puzzle)

**Input:**  
The first line of input is  T denoting the number of testcases.Then each of the T lines contains two positive integer N and K where 'N' is the number of eggs and 'K' is number of floor in building.

**Output:**  
For each test case, print a single line containing one integer the minimum number of attempt you need in order find the critical floor.

**Constraints:**  
1<=T<=30  
1<=N<=10  
1<=K<=50

**Example:**  
**Input:**  
2  
2 10  
3 5

**Output:**  
4  
3

/\* Function to get minimum number of trials needed in worst

  case with n eggs and k floors \*/

int eggDrop(int n, int k)

{

    /\* A 2D table where entery eggFloor[i][j] will represent minimum

       number of trials needed for i eggs and j floors. \*/

    int eggFloor[n+1][k+1];

    int res;

    int i, j, x;

    // We need one trial for one floor and0 trials for 0 floors

    for (i = 1; i <= n; i++)

    {

        eggFloor[i][1] = 1;

        eggFloor[i][0] = 0;

    }

    // We always need j trials for one egg and j floors.

    for (j = 1; j <= k; j++)

        eggFloor[1][j] = j;

    // Fill rest of the entries in table using optimal substructure

    // property

    for (i = 2; i <= n; i++)

    {

        for (j = 2; j <= k; j++)

        {

            eggFloor[i][j] = INT\_MAX;

            for (x = 1; x <= j; x++)

            {

                res = 1 + max(eggFloor[i-1][x-1], eggFloor[i][j-x]);

                if (res < eggFloor[i][j])

                    eggFloor[i][j] = res;

            }

        }

    }

    // eggFloor[n][k] holds the result

    return eggFloor[n][k];

}

## [Optimal Strategy for a Game](https://practice.geeksforgeeks.org/problems/optimal-strategy-for-a-game/0)

Company Tags [Amazon](https://practice.geeksforgeeks.org/company/Amazon/)  [Google](https://practice.geeksforgeeks.org/company/Google/)  [Hike](https://practice.geeksforgeeks.org/company/Hike/)  [Linkedin](https://practice.geeksforgeeks.org/company/Linkedin/)  [Microsoft](https://practice.geeksforgeeks.org/company/Microsoft/)  [Salesforce](https://practice.geeksforgeeks.org/company/Salesforce/)

* [**Problems**](https://practice.geeksforgeeks.org/problems/optimal-strategy-for-a-game/0#problems)

You are given an array **A of size N**. The array contains integers and is of **even length**. The elements of the array represent N **coin**of **values V1, V2, ....Vn**. You play against an opponent in an **alternating**way.

In each **turn**, a player selects either the **first or last coin** from the **row**, removes it from the row permanently, and **receives the value** of the coin.

You need to determine the **maximum possible amouint of money**you can win if you **go first**.

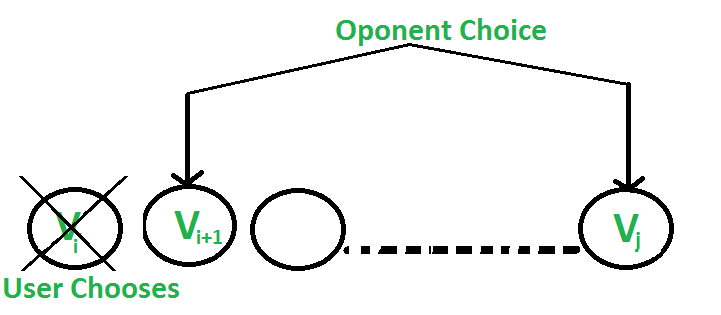
**Input:**  
The first line of input contains **T** denoting the number of testcases. T testcases follow. Each testcase contains two lines of input. The first line contains **N** denoting the size of the array. The second line contains N elements of the array.

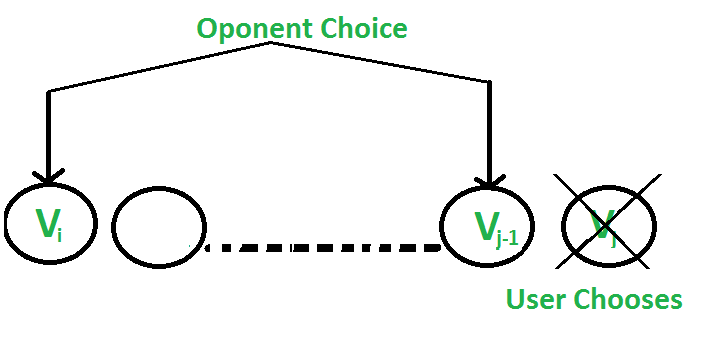
**Output:**  
For each testcase, in a new line, print the **maximum amout**.

**Constraints:**  
1 <= T <= 100  
2 <= N <= 100  
1 <= Ai <= 106

**Examples:  
Input:**  
2  
4  
5 3 7 10  
4  
8 15 3 7  
**Output:**  
15  
22

**Explanation:**  
**Testcase1:** The user collects maximum value as 15(10 + 5)  
**Testcase2:**The user collects maximum value as 22(7 + 15)

There are two choices:  
**1.** The user chooses the ith coin with value Vi: The opponent either chooses (i+1)th coin or jth coin. The opponent intends to choose the coin which leaves the user with minimum value.  
i.e. The user can collect the value Vi + min(F(i+2, j), F(i+1, j-1) )  
[](https://media.geeksforgeeks.org/wp-content/cdn-uploads/coinGame1.png)

**2.** The user chooses the jth coin with value Vj: The opponent either chooses ith coin or (j-1)th coin. The opponent intends to choose the coin which leaves the user with minimum value.  
i.e. The user can collect the value Vj + min(F(i+1, j-1), F(i, j-2) )  
[](https://media.geeksforgeeks.org/wp-content/cdn-uploads/coinGame21.png)

F(i, j) represents the maximum value the user can collect from

i'th coin to j'th coin.

F(i, j) = Max(Vi + min(F(i+2, j), F(i+1, j-1) ),

Vj + min(F(i+1, j-1), F(i, j-2) ))

Base Cases

F(i, j) = Vi If j == i

F(i, j) = max(Vi, Vj) If j == i+1

int solve(vector<int> &A, int n){

if(n==1) return A[0];

vector<vector<int>>dp(n,vector<int>(n));

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

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

int x=(i+2<=j)?dp[i+2][j]:0;

int y=(i+1<=j-1)?dp[i+1][j-1]:0;

int z=(i<=j-2)?dp[i][j-2]:0;

dp[i][j]=max(A[i]+min(x,y), A[j]+min(y,z));

}

}

return dp[0][n-1];

}

## [Shortest Common Supersequence](https://practice.geeksforgeeks.org/problems/shortest-common-supersequence/0)

Length of the shortest supersequence = (Sum of lengths of given two strings) - (Length of LCS of two given strings)

Company Tags [Microsoft](https://practice.geeksforgeeks.org/company/Microsoft/)

* [**Problems**](https://practice.geeksforgeeks.org/problems/shortest-common-supersequence/0#problems)

Given two strings **str1** and **str2**, find the length of the smallest string which has both, str1 and str2 as its sub-sequences.  
**Note:**str1 and str2 can have both uppercase and lowercase letters.

**Input:**  
The first line of input contains an integer T denoting the number of test cases.Each test case contains two space separated strings.

**Output:**  
For each testcase, in a new line, output the length of the required string.

**Constraints:**  
1 <= T <= 100  
1<= |str1|, |str2| <= 100

**Example:  
Input:**  
2  
abcd xycd  
efgh jghi  
**Output:**  
6  
6

int lcs(string &A, string &B){

int lenA=A.size();

int lenB=B.size();

if(lenA==0 || lenB==0) return 0;

vector<vector<int>>dp(lenA+1, vector<int>(lenB+1,0));

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

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

if(A[i-1]==B[j-1])

dp[i][j]=dp[i-1][j-1]+1;

else

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

}

}

return dp[lenA][lenB];

}

int lcSS(string &A, string &B){

return A.size()+B.size()-lcs(A,B);

}

# **Divide and Conquer**

* x&(x-1) : clears lowest set bit
* x&~(x-1) : extracts lowest set bit; remaining bits are 0
* x|(x+1) : sets lowest clear bit
* x|~(x+1) : extracts lowest clear bit; remaining bits are 1

## [Find the element that appears once in sorted array](https://practice.geeksforgeeks.org/problems/find-the-element-that-appears-once-in-sorted-array/0)

Company Tags [Amazon](https://practice.geeksforgeeks.org/company/Amazon/)  [D-E-Shaw](https://practice.geeksforgeeks.org/company/D-E-Shaw/)  [Ola Cabs](https://practice.geeksforgeeks.org/company/Ola%20Cabs/)

* [**Problems**](https://practice.geeksforgeeks.org/problems/find-the-element-that-appears-once-in-sorted-array/0#problems)

Given a sorted array**A**, size **N**, of integers; every element appears twice except for one. Find that element that appears once in array.

**Input:**  
The first line of input consists of **T**, the number of the test cases. T testcases follow. Each testcase contains two lines of input.  
The first line of each test case contains the size of the array, and the second line has the elements of the array.

**Output:**  
For each testcase, in a new line, print the number that appears only once in the array.

**Constraints:**  
1 ≤ T ≤ 100  
1 ≤ N ≤ 107  
0 ≤ A[i] ≤ 1017

**Example:**  
**Input:**  
1  
11  
1 1 2 2 3 3 4 50 50 65 65  
**Output:**  
4

#### Sol:

int singleOccurence(vector<int>&A, int n){

if(n<1) return -1;

if(n==1) return A[0];

int low=0, high=n-1;

while(low<high){

int mid=(low+high)/2;

if(mid%2==0){

if(A[mid]==A[mid+1])

low=mid+2;

else

high=mid;

}

else{

if(A[mid]==A[mid-1])

low=mid+1;

else

high=mid;

}

}

if(low==high) return A[low];

return -1;

}

# [Search in a Rotated Array](https://practice.geeksforgeeks.org/problems/search-in-a-rotated-array/0)

int findStart(vector<int>&A){

int n=A.size();

if(n<1) return -1;

int low=0, high=n-1;

while(low<high){

int mid=(low+high)/2;

if(A[mid]>A[high])

low=mid+1;

else

high=mid;

}

if(low==high) return low;

return -1;

}

int binSearch(vector<int> &A, int low, int high, int x){

while(low<=high){

int mid=low+(high-low)/2;

if(A[mid]==x) return mid;

if(A[mid]<x) low=mid+1;

else high=mid-1;

}

return -1;

}

class Solution {

public:

int search(vector<int>& A, int x) {

int n=A.size();

if(n<1) return -1;

int start=findStart(A);

if(start==-1) return -1;

int res=binSearch(A, 0, start-1, x);

if(res==-1)

res=binSearch(A, start, n-1, x);

return res;

}

};

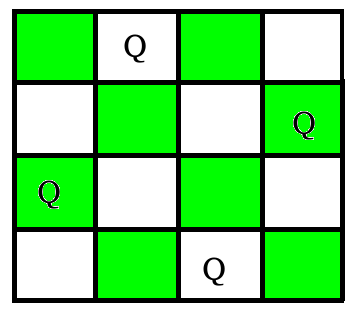
1. [Binary Search](https://practice.geeksforgeeks.org/problems/binary-search/1)
2. [Sum of Middle Elements of two sorted arrays](https://practice.geeksforgeeks.org/problems/sum-of-middle-elements-of-two-sorted-arrays/0)
3. [Quick Sort](https://practice.geeksforgeeks.org/problems/quick-sort/1)
4. [Merge Sort](https://practice.geeksforgeeks.org/problems/merge-sort/1)
5. [K-th element of two sorted Arrays](https://practice.geeksforgeeks.org/problems/k-th-element-of-two-sorted-array/0)

# **Backtracking**

## [N-Queen Problem](https://practice.geeksforgeeks.org/problems/n-queen-problem/0)

Company Tags [Accolite](https://practice.geeksforgeeks.org/company/Accolite/)  [Amazon](https://practice.geeksforgeeks.org/company/Amazon/)  [Amdocs](https://practice.geeksforgeeks.org/company/Amdocs/)  [D-E-Shaw](https://practice.geeksforgeeks.org/company/D-E-Shaw/)  [MAQ Software](https://practice.geeksforgeeks.org/company/MAQ%20Software/)  [Twitter](https://practice.geeksforgeeks.org/company/Twitter/)  [Visa](https://practice.geeksforgeeks.org/company/Visa/)

* [**Problems**](https://practice.geeksforgeeks.org/problems/n-queen-problem/0#problems)

The n-queens puzzle is the problem of placing n queens on an n×n chessboard such that no two queens attack each other. Given an integer n, print all distinct solutions to the n-queens puzzle. Each solution contains distinct board configurations of the n-queens’ placement, where the solutions are a permutation of [1,2,3..n] in increasing order, here the number in the ith place denotes that the ith-column queen is placed in the row with that number. For eg below figure represents a chessboard [3 1 4 2].  
  
  
**Input:**  
The first line of input contains an integer **T** denoting the no of test cases. Then T test cases follow. Each test case contains an integer n denoting the size of the chessboard.

**Output:**  
For each test case, output your solutions on one line where each solution is enclosed in square brackets '[', ']' separated by a space . The solutions are permutations of {1, 2, 3 …, n} in increasing order where the number in the ith place denotes the ith-column queen is placed in the row with that number, if no solution exists print -1.

**Constraints:**  
1 <= T <= 10  
1 <= n <= 10

**Example:  
Input**  
2  
1  
4  
**Output:**  
[1 ]  
[2 4 1 3 ] [3 1 4 2 ]

Sol:

#include <iostream>

#include<bits/stdc++.h>

using namespace std;

vector<vector<int>> res;

struct mycomp{

bool operator()(const vector<int>&A, const vector<int> &B){

int n=A.size();

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

if(A[i]!=B[i]) return A[i]<B[i];

}

}myobj;

void printVector(vector<vector<int>>&res){

int row=res.size();

if(row<1){

cout<<-1;

return;

}

int col=res[0].size();

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

cout<<"[";

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

cout<<res[i][j]<<" ";

cout<<"] ";

}

}

bool isSafe(vector<vector<bool>> &sol, int row, int col, int n){

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

if(sol[row][i]==1)

return false;

for(int i=row, j=col; i>=0 && j>=0; i--, j--)

if(sol[i][j]==1)

return false;

for(int i=row, j=col; i<n && j>=0; i++, j--)

if(sol[i][j]==1)

return false;

return true;

}

void solveNQueensUtil(vector<vector<bool>> &sol, int col, int n){

if(col>=n){

//print sol

vector<int> tmp;

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

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

if(sol[i][j])

tmp.push\_back(j+1);

}

res.push\_back(tmp);

return;

}

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

if(isSafe(sol, i, col, n)){

sol[i][col]=1;

solveNQueensUtil(sol, col+1, n);

sol[i][col]=0;

}

}

}

void solveNQueens(int n){

res.clear();

vector<vector<bool>>sol;

sol.resize(n, vector<bool>(n,0));

solveNQueensUtil(sol, 0, n);

sort(res.begin(), res.end(), myobj);

printVector(res);

cout<<endl;

}

int main() {

//code

int t,n;

cin>>t;

while(t--){

cin>>n;

solveNQueens(n);

}

return 0;

}

1. [Solve the Sudoku](https://practice.geeksforgeeks.org/problems/solve-the-sudoku/0)
2. [Rat in a Maze Problem](https://practice.geeksforgeeks.org/problems/rat-in-a-maze-problem/1)
3. [Word Boggle](https://practice.geeksforgeeks.org/problems/word-boggle/0)
4. [Generate IP Addresses](https://practice.geeksforgeeks.org/problems/generate-ip-addresses/1)

# **Bit Magic**

## [Find first set bit](https://practice.geeksforgeeks.org/problems/find-first-set-bit/0)

Company Tags [Amazon](https://practice.geeksforgeeks.org/company/Amazon/)

* [**Problems**](https://practice.geeksforgeeks.org/problems/find-first-set-bit/0#problems)

Given an integer an **N.** The task is to print the position of **first set bit found from right side** in the binary representation of the number.

**Input:**  
The first line of the input contains an integer T, denoting the number of test cases. Then T test cases follow. The only line of the each test case contains an integer N.

**Output:**  
For each test case print in a single line an integer denoting the position of the first set bit found form right side of the binary representation of the number.**If there is no set bit print "0"**.  
  
**Constraints:**  
1 <= T <= 200  
0 <= N <= 106  
  
**Example:**  
**Input:**  
2  
18  
12

**Output:**  
2  
3

#### Sol:

unsigned int getFirstSetBitPos(int n)

{

    return log2(n & ~(n-1)) + 1;

}

## [Rightmost different bit](https://practice.geeksforgeeks.org/problems/rightmost-different-bit/0)

* [**Problems**](https://practice.geeksforgeeks.org/problems/rightmost-different-bit/0#problems)

Given two numbers **M** and **N**. The task is to find the position of **rightmost different** bit in binary representation of numbers.

**Input:**  
The input line contains T, denoting the number of testcases. Each testcase follows. First line of each testcase contains two space separated integers M and N.

**Output:**  
For each testcase in new line, print the position of rightmost different bit in binary representation of numbers. If both M and N are same then print **-1** in this case.

**Constraints:**  
1 <= T <= 100  
1 <= M <= 103  
1 <= N <= 103

**Example:**  
**Input:**  
2  
11 9  
52 4

**Output:**  
2  
5

// Function to find the position of

// rightmost set bit in 'n'

int getRightMostSetBit(int n)

{

    return log2(n & -n) + 1;

}

// Function to find the position of

// rightmost different bit in the

// binary representations of 'm' and 'n'

int posOfRightMostDiffBit(int m, int n)

{

    // position of rightmost different

    //  bit

    return getRightMostSetBit(m ^ n);

}

1. [Check whether K-th bit is set or not](https://practice.geeksforgeeks.org/problems/check-whether-k-th-bit-is-set-or-not/0)
2. [Toggle bits given range](https://practice.geeksforgeeks.org/problems/toggle-bits-given-range/0)
3. [Set kth bit](https://practice.geeksforgeeks.org/problems/set-kth-bit/0)
4. [Power of 2](https://practice.geeksforgeeks.org/problems/power-of-2/0)
5. [Bit Difference](https://practice.geeksforgeeks.org/problems/bit-difference/0)
6. [Rotate Bits](https://practice.geeksforgeeks.org/problems/rotate-bits/0)
7. [Swap all odd and even bits](https://practice.geeksforgeeks.org/problems/swap-all-odd-and-even-bits/0)
8. [Count total set bits](https://practice.geeksforgeeks.org/problems/count-total-set-bits/0)
9. [Longest Consecutive 1’s](https://practice.geeksforgeeks.org/problems/longest-consecutive-1s/0)
10. [Sparse Number](https://practice.geeksforgeeks.org/problems/number-is-sparse-or-not/0)
11. [Alone in a couple](https://practice.geeksforgeeks.org/problems/alone-in-couple/0)
12. [Maximum subset XOR](https://practice.geeksforgeeks.org/problems/maximum-subset-xor/1)

# Design