8.1 Introduction Array.

Array: "Array is a list of variables of similar type."
dataType arrayName[size];

Ex: {1,2,3}   {a,b,c,d}
int array[4] = {40,45,50,55};
int   = 4 Bytes
Array = 4 X 4 bytes = 16 bytes

Q. Find Maximum and Minimum Number in Array ?

```
#include<iostream>
#include<limits.h> // new header file
using namespace std;

int main(){
    int n;
    cin>>n;
    int arr[n];

    for(int i=0; i<n; i++){
        cin>>arr[i];
    }
    int maxNo=INT_MIN;  // new header file used
    int minNo=INT_MAX;  // new header file used
    for(int i=0; i<n; i++){
        maxNo=max(maxNo,arr[i]);  //Buillting Function used 'max'
        minNo=min(minNo,arr[i]);  //Buillting Function used 'min'
    }
    cout<<"MaxNo."<<maxNo<<"\tMinNo."<<minNo<<endl;
    return 0;
}
```

8.2 Searching in Array.

2.1 Linear Search.

O (n) time complexity.(Normal)

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

int linearSearch(int arr[], int n, int key){
    for(int i=0; i<n; i++){
        if(arr[i]==key){
            return i;
        }
    }
    return -1;  //condition not match then output -1 print.
}
int main()
{
    int n;
    cin>>n;

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

    cout<<linearSearch(arr,n,key)<<endl;
    return 0;
}
```

2. Binary Search Date:28-02-22

After first iteration, length of Array = n
After second iteration, length of Array = n/2
After third iteration, length of Array = (n/2)/2
After K iteration, length of Array = $n/2^k$

Let the length of array become 1 after K iterations
= $n/2^k = 1$
= $n = 2^k$
= $\log_2 (n) = \log_2 (2^k)$
= $\log_2 (n) = K (\log_2)^2$
= $K = (\log_2)^2$

Time Complexity $O(\log_2)^n$

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

int binarySearch(int arr[], int n, int key){
    int s=0;
    int e=n;
    while(s<=e){
        int mid=(s+e)/2;

        if(arr[mid]==key){
            return mid;
        }
        else if(arr[mid]>key){
            e=mid-1;
        }
        else{
            s=mid+1;
        }
    }
    return -1;
}
int main(){
    int n;
    cin>>n;

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

    cout<<binarySearch(arr,n,key)<<endl;
    return 0;
}
```

8.3 Sorting in Array. Date:28-02-22

3.1 Selection Short
"Find the minimum element in unsorted array and swap it with element at beginning."
Example:
```
   12,45,23,51,19,8 is Given Shorting Array
   8,45,23,51,19,12
   8,12,23,51,19,45
   8,12,19,51,23,45
   8,12,19,23,51,45
   8,12,19,23,45,51
```
Final Shorting Array is 8,12,19,23,45,51
```
#include<iostream>
using namespace std;
int main()
{
    int n;
    cin>>n;
    int arr[n];
    for(int i=0; i<n; i++){
        cin>>arr[i];
    }

    for(int i=0; i<n-1; i++){
        for(int j=i+1; j<n; j++){
            if(arr[j]<arr[i]){
                int temp=arr[j];
                arr[j]=arr[i];
                arr[i]=temp;
            }
        }
    }
    for(int i=0; i<n; i++){
        cout<<arr[i]<<" ";
    }
    cout<<endl;
    return 0;
}
```

3.2. Bubble Short.
"Repeatedly swap two adjacent elements if they are in wrong order."

ith iteration = n-i
```
1.12,45,23,51,19,8   3. 12,23,19,8,45,51
   12,45,23,51,19,8      12,23,19,8,45,51
   12,23,45,51,19,8      12,19,23,8,45,51
   12,23,45,51,19,8      12,19,8,23,45,51
   12,23,45,19,51,8
   12,23,45,19,8,51   4. 12,19,8,23,45,51
                         12,19,8,23,45,51
2. 12,23,45,19,8,51      12,8,19,23,45,51
   12,23,45,19,8,51
   12,23,45,19,8,51   5. 12,8,19,23,45,51
   12,23,19,45,8,51      8,12,19,23,45,51
   12,23,19,8,45,51
```
Final Shorting array is 8,12,19,23,45,51
```
#include<iostream>
using namespace std;
```

```
int main()
{
    int n;
    cin>>n;
    int arr[n];
    for(int i=0; i<n; i++){
        cin>>arr[i];
    }
    int counter=1;
    while(counter<n-1){
        for(int i=0; i<n-counter; i++){
            if(arr[i]>arr[i+1]){
                int temp=arr[i];
                arr[i]=arr[i+1];
                arr[i+1]=temp;
            }
        }
        counter++;
    }
    for(int i=0; i<n; i++){
        cout<<arr[i]<<" ";
    }
    cout<<endl;
    return 0;
}
```

3.3 Insertion Shorting.

```
12,45,23,51,19,8
12,45,23,51,19,8
12,23,45,51,19,8
12,23,45,51,19,8
12,19,23,45,51,8
```

8,12,19,23,45,51 Final Shorting Array is : 8,12,19,23,45,51

```
#include<iostream>
using namespace std;
int main()
{
    int n;
    cin>>n;
    int arr[n];
    for(int i=0; i<n; i++){
        cin>>arr[i];
    }
    for(int i=1; i<n; i++){
        int current = arr[i];
        int j = i-1;
        while(arr[j]>current && j>=0)
        {
            arr[j+1]=arr[j];
            j--;
        }
        arr[j+1]=current;
    }
    for(int i=0; i<n; i++){
        cout<<arr[i]<<" ";
    }
    cout<<endl;
    return 0;
}
```

8.4 Array challenges

Problem 1:

Given an array a[] of size n. for every 'i' from 0 to n-1.

Approach:

1. Keep a variable mx which stores the maximum till 'i' element.
2. Iteration over the array and update.

   mx = max(mx, a[i])

Output: max(a[0],a[1],....a[i]).

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

int main()
{
    int mx = -1999999;
    int n;
    cin>>n;

    int a[n];
    for(int i=0; i<n; i++){
        cin>>a[i];
    }
    for(int i=0; i<n; i++){
        mx=max(mx, a[i]);
        cout<<mx<<endl;
    }
    return 0;
}
```

Sub-array v/s sub-sequence array.

Sub-Array:

Continuous part of the array.

Array with nth element = nc_r+n = n*(n+1)/2.

Sub-Sequence Array:

A Sub-sequence is a sequence that can be derived an array by selection zero or more element, without changing the order of the remaining elements.

Array with nth element = 2^n.

Problem 2:

Given an array a[] of size n. **Output**: sum of each sub-array of the given array.

```

#include<iostream>
using namespace std;
int main()
{
    int n;
    cin>>n;
    int a[n];
    for(int i=0; i<n; i++){
        cin>>a[i];
    }
    int curr=0;
    for(int i=0; i<n; i++)
    {
        curr=0;
        for(int j=i; j<n; j++){
            curr += a[j];
            cout<<a[j]<<" ";
        }
    }
    return 0;
}
```

Important Question: longest Arithmetic sub-Array

(Google kick-start)

Problem:

An arithmetic array is an array that contains at least two integers and the differences between consecutive integers are equal. For example, [9, 10], [3, 3, 3], and [9, 7, 5, 3] are arithmetic arrays, while [1, 3, 3, 7], [2, 1, 2], and [1, 2, 4] are not arithmetic arrays.

Sarasvati has an array of **N** non-negative integers. The i-th integer of the array is $\mathbf{A_i}$. She wants to choose a contiguous arithmetic sub array from her array that has the maximum length. Please help her to determine the length of the longest contiguous arithmetic sub array.

Input:

The first line of the input gives the number of test cases, **T. T** test cases follow. Each test case begins with a line containing the integer **N**. The second line contains N integers. The i-th integer is $\mathbf{A_i}$.

Output:

For each test case, output one line containing **Case #x: y**, where **x** is the test case number (starting from 1) and **y** is the length of the longest contiguous arithmetic subarray.

Constraints:

Time limit: 20 seconds per test set.
Memory limit: 1GB.
$1 \le T \le 100$.
$0 \le A_i \le 10^9$
Test Set 1
$2 \le N \le 2000$.
Test Set 2
$2 \le N \le 2 \times 10^5$ for at most 10 test cases.
For the remaining cases, $2 \le N \le 2000$.

Solution:

Constraints Analysis

1 sec = 10^8 operations
20 sec = $2x10^9$ operations

Intuition:

We have to loop over the array and find the answer.

Steps:

1. While iterating in the array we need to maintain the following variables,
   a. Previous common difference (pd) - To compare it with current common difference **`(a[i] - a[i-1]).`**
   b. Current arithmetic subarray length (curr) - It denotes the arithmetic subarray length including **`a[i].`**
   c. Maximum arithmetic subarray length (ans) - It denotes the max. Arithmetic subarray length till **`a[i].`**

2. While iterating, there will be two cases,
   a. **pd = a[i] - a[i-1]**
      i. Increase curr by 1
      ii. **`ans = max(ans, curr)`**

3. After loop ends, output the answer. (stored in ans variable).

```cpp
#include<iostream>
using namespace std;
//longest Arithematic sub-Array
int main()
{
    int n;
    cin>>n;
    int a[n];
    for(int i=0; i<n; i++){
        cin>>a[i];
    }
    int ans = 2;
    int pd = a[1]-a[0];
    int j=2;
    int curr = 2;

    while(j<n)
    {
        if(pd == a[j] - a[j-1])
        {
            curr++;
        }
        else{
            pd = a[j] - a[j-1];
            curr = 2;
        }
        ans = max(ans, curr);

        j++;
    }
    cout<<ans<<endl;
    return 0;
}
```

Arrays Challenge-Record Breaker

(Google kickstart)

Problem:

Isyana is given the number of visitors at her local theme park on **N** consecutive days. The number of visitors on the i-th day is $\mathbf{V_i}$. A day is record breaking if it satisfies both of the following conditions:

- The number of visitors on the day is strictly larger than the number of visitors on each of the previous days.
- Either it is the last day, or the number of visitors on the day is strictly larger than the number of visitors on the following day.

Note that the very first day could be a record breaking day! Please help Isyana find out the number of record breaking days.

Input:

The first line of the input gives the number of test cases, **T. T** test cases follow. Each test case begins with a line containing the integer **N**. The second line contains **N** integers. The i-th integer is $\mathbf{V_i}$.

Output :

For each test case, output one line containing Case #x: y, where x is the test case number (starting from 1) and y is the number of record breaking days.

Constraints
Time limit: 20 seconds per test set.
Memory limit: 1GB.
1 ≤ T ≤ 100.
0 ≤ Vi ≤ 2 × 105 .
Test set 1
1 ≤ N ≤ 1000.
Test set 2
1 ≤ N ≤ 2 × 105 for at most 10 test cases.
For the remaining cases, 1 ≤ N ≤ 1000.

Solution :

Constraints Analysis :

1 sec = 108 operations

20 sec = 2x109 operations

Brute Force Approach:

Iterate over all the elements and check if it is record breaking day or not.

Note: To check if a[i] is a record breaking day, we have to iterate over a[0], a[1],...., a[i-1].

Time complexity for this operation: O(n)

Overall Time Complexity: $O(n^2)$

Optimised Approach:

Intuition: If we can optimise step (1), then we can optimise our overall solution.

For step (1): We need to check if a[i] > { a[i-1], a[i-2],..., a[0] }, which is same as

a[i] > max(a[i-1], a[i-2],..., a[0])

For this, we will keep a variable mx, which will store the maximum value till a[i]. Then we just need to check, a[i] > mx

a[i] > a[i+1] , { if i+1 < n }

and update mx, mx = max(mx, a[i])

So step (1) time complexity reduces to O(1).

Overall time complexity: O(n) Code

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

// record Breaker array
int main()
{
    int n;
    cin>>n;

    int a[n+1];
    a[n]= -1;

    for(int i=0; i<n; i++){
        cin>>a[i];
    }
    if(n == 1){
        cout<<"1"<<endl;
        return 0;
    }

    int ans = 0;
    int mx = -1;

    for(int i=0; i<n; i++){
        if(a[i]>mx && a[i]>a[i+1]){
            ans++;
        }
        mx = max(mx, a[i]);
    }
    cout<<ans<<endl;
    return 0;
}
```

8.5 Array – Q's asked by top MNC's.

Arrays Challenge-First Repeating Element

(Amazon, Oracle)

Problem:

Given an array **arr**[] of size **N**. The task is to find the first repeating element in an array of integers, i.e., an element that occurs more than once and whose index of first occurrence is smallest.

Constraints

$1 <= N <= 10^6$

$0 <= Ai <= 10^6$

Example

Input:

7

1 5 3 4 3 5 6

Output: 2

Explanation:

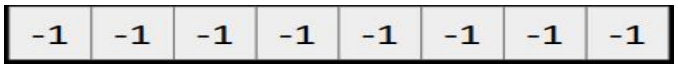
5 is appearing twice and its first appearance is at index 2 which is less than 3 whose first occurring index is 3.

Solution: Base idea:

To check if an element is repeating, we maintain an array idx[], which stores the first occurrence (index) of a particular element a[i].

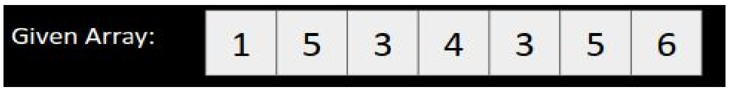
Steps:

1. Initialise the idx[] with -1, and minidx with INT_MAX.



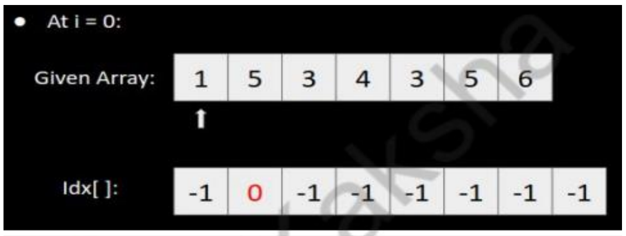
-1	-1	-1	-1	-1	-1	-1	-1

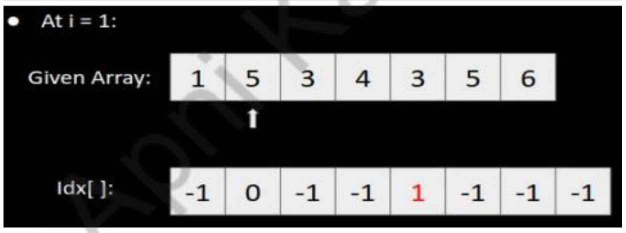
2. Keep updating idx[], while traversing the given array.

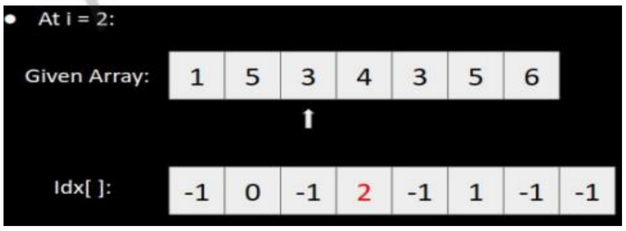


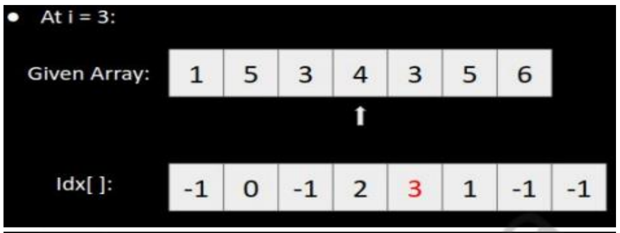
Given Array:							
	1	5	3	4	3	5	6

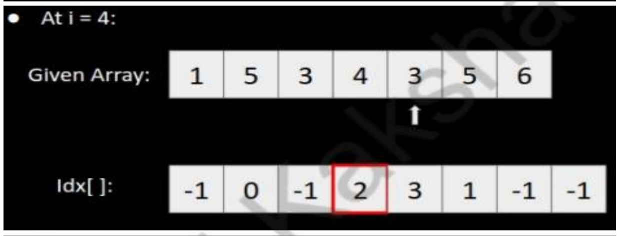
Iterations:

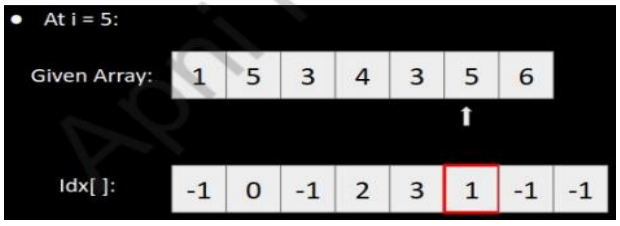


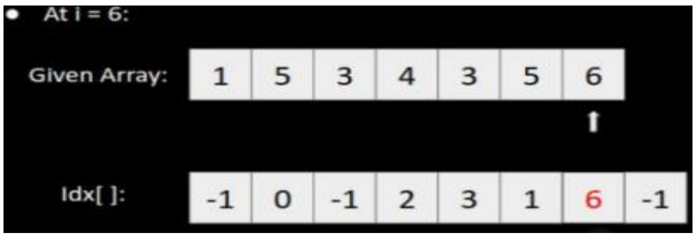












```
#include<bits/stdc++.h>
using namespace std;
int main() {
    int n;
    cin>>n;
    int a[n];
    for(int i=0; i<n; i++){
        cin>>a[i];
    }
    const int N = 1e6+2;
    int idx[N];
    for(int i=0; i<N;i++){
        idx[i] = -1;
    }
    int minidx = INT_MAX;
    for(int i=0; i<n; i++)
    {
        if(idx[a[i]] != -1)
        {
            minidx = min(minidx, idx[a[i]]);
        }
        else
        {
           idx[a[i]] = i;
        }
    }
    if(minidx == INT_MAX)
    {
        cout<<"-1"<<endl;
    }
    else
    {
        cout << minidx +1 << endl;
    }
    return 0;
}
```

Arrays Challenge-Subarray with given Sum

(Google, Amazon, Facebook, Visa)

Problem:

Given an unsorted array **A** of size **N** of non-negative integers, find a continuous subarray which adds to a given number **S**.

Constraints

$1 <= N <= 10^5$

$0 <= A_i <= 10^{10}$

Example

Input:

N = 5, S = 12

A[] = {1,2,3,7,5}

Output**: 2 4**

Explanation: The sum of elements from 2nd position to 4th position is 12.

Solution:

Brute Force Solution

- Find sum of all possible sub-arrays. If any of the sum equates to S, output the starting and ending index of the sub-array.

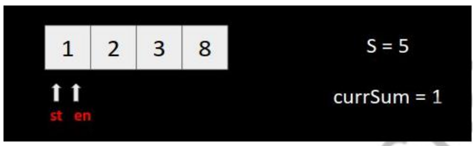
  Time Complexity: **O(n^2)**

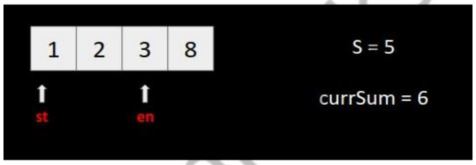
Optimized Approach

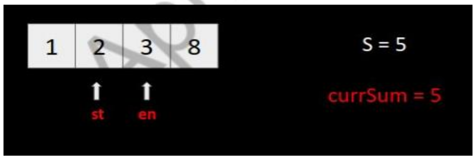
Steps:

1. Keep the pointers st and en, and a variable currSum that stores the sum from st to en.
2. Initialize st = 0, en = 0
3. Increment en till currSum + a[en + 1] > S
4. When 3rd condition occurs, start increasing st until currSum <= S.
5. Whenever the condition (currSum = S) is satisfied, store st and en and BREAK from the loop.

Iterations:







Code:

```
#include<bits/stdc++.h>
using namespace std;
/*  subarray with given sum */
int main()
{
    int n, s;
    cin >>n>>s;
    int a[n];
    for(int i=0; i<n; i++){
        cin>>a[i];
    }

    int i=0, j=0, st=-1, en=-1, sum=0;
    while(j<n && sum+a[j]<=s)
    {
        sum += a[j];
        j++;
    }
    if(sum==s){
        cout<<i+1<<""<<j<<endl;
        return 0;
    }
    while(j<n)
    {
        sum += a[j];
        while(sum>s){
            sum -= a[i];
            i++;
        }
        if(sum==s){
            st = i+1;
            en = j+1;
            break;
        }
        j++;
    }
    cout<<st<<" "<<en<<endl;
    return 0;
}
```

Arrays Challenge - Smallest Positive Missing Number

(Amazon, Samsung, Snapdeal, Accolite)

Problem

Find the smallest positive missing number in the given array.

Example: [0, -10, 1, 3, -20], Ans = 2

Intuition;

If in O(1), we can tell if an element is present in an array, then our task will be simpler.

For that, we will maintain a Check[ ] array, that will if an element x is present in the array or not.

It will be of boolean type as we only need to check the presence or absence of the number.

Steps to Solve:

1. Build the Check[ ] array initialized with False at all indices.
2. By iterating over the array and marking non-negative a[i] as true i.e.

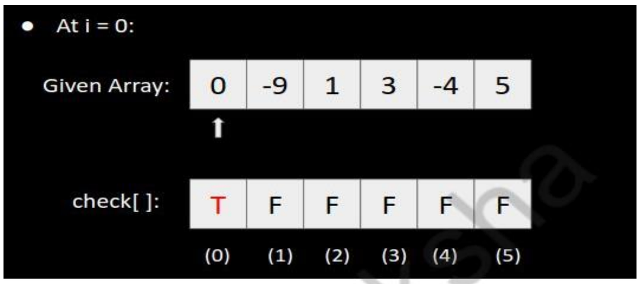
```
        if(a[i] >= 0)
          check[a[i]] = True
```

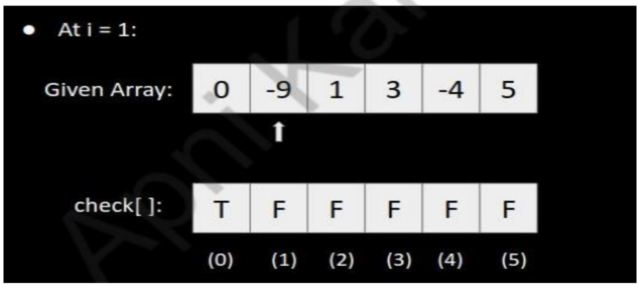
3. Iterate in the Check[ ] from i=1, BREAK the loop when you find check[i] =False and store that i in the ans variable.
4. Output the ans.

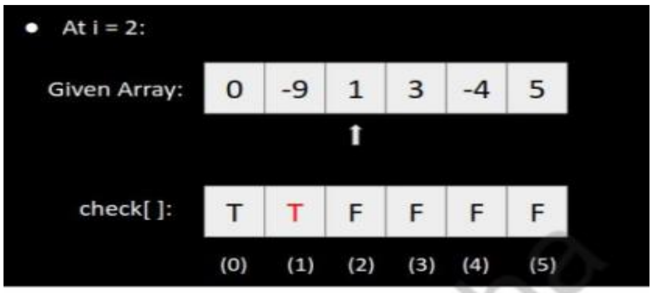
Example:

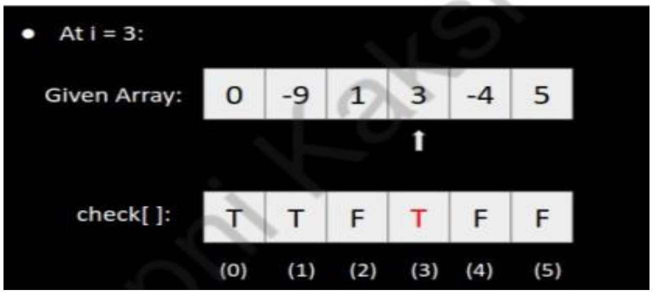
Given Array: [0, -9, 1, 3, -4, 5]

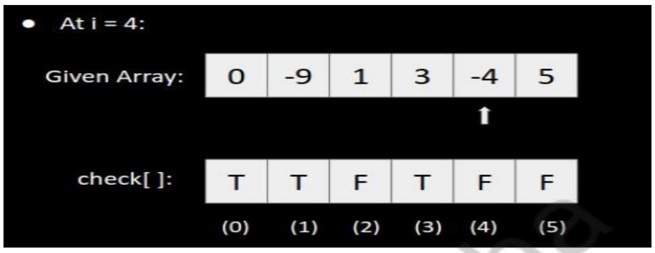
Iterations:

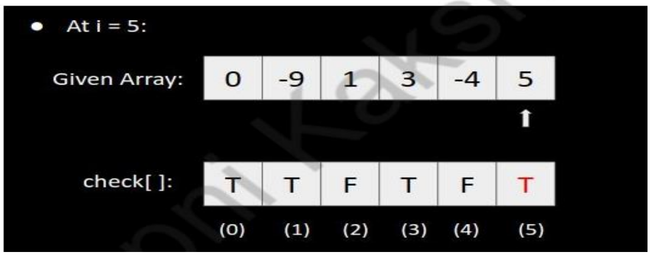


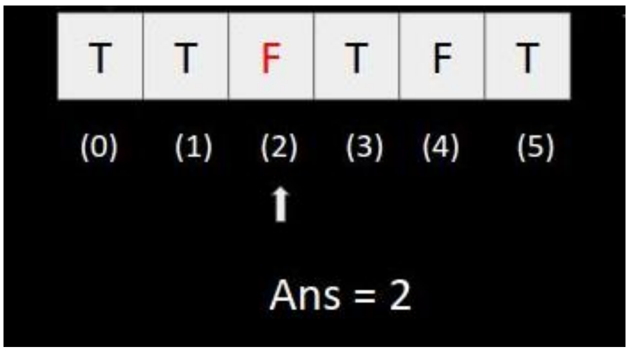












Program:

```cpp
#include<bits/stdc++.h>
using namespace std;
/* Smallest positive missing number.   */
int main()
{
    int n;
    cin>>n;
    int a[n];
    for(int i=0; i<n; i++){
        cin>>a[i];
    }
    const int N = 1e6 + 2;
    bool check[N];
    for(int i=0; i<n; i++){
        check[i]= false;
    }
    for(int i=0; i<n; i++){
        if(a[i] >= 0){
            check[a[i]] = 1;
        }
    }
    int ans = -1;
    for(int i=1; i<N; i++){
        if(check[i] == false){
            ans = i;
            break;
        }
    }
    cout<<ans<<endl;
    return 0;
}
```

8.6 Sub-Array:

"Subarray is a Continuous part of the array". **Ex**:- {-2,0,3,7,11}
Array with n^th element = $^{n}c_{r}$+n = n*(n+1)/2.

Q.Given sub array print . Input = {-1,4,7,2}

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

int main()
{
    int n;
    cin>>n;
    int arr[n];
    for(int i=0; i<n; i++)
    {
        cin>>arr[i];
    }
    for (int i=0; i<n; i++)
    {
        for(int j=i; j<n; j++)
        {
            for(int k=i; k<=j; k++)
            {
                cout<<arr[k]<<" ";
            }
            cout<<endl;
        }
    }
    return 0;
}
```

Question:- Find the subarray is an array which has maximum sum.

1. Brute Force:

Idea: For each subarray arr[i..j], calculate its sum. Time Complexity : **O(N^3).** Space Complexity : **O(1)**

```
#include<bits/stdc++.h>
using namespace std;
int main()
{   int n, sum=0;
    int maxSum=INT_MIN;
    cin>>n;
    int arr[n];
    for(int i=0; i<n; i++) {
        cin>>arr[i];
    }
    for (int i=0; i<n; i++) {
        for(int j=i; j<n; j++) {
            for(int k=i; k<=j; k++) {
                sum += arr[k];
            }
            maxSum=max(maxSum, sum);
        }
    }
    cout<<maxSum<<endl;
    return 0;
}
```

2. Cumulative Sum approach:

Idea: For each subarray arr[i..j], calculate its sum. Time Complexity: **$O(N^2)$**Space Complexity**: O(N)**

```cpp
#include<bits/stdc++.h>
using namespace std;
/*  Maximum Sum Subarray with Cumulative Sum approach:   */
int main()
{
    int n;
    cin>>n;
    int arr[n];
    for(int i=0; i<n; i++){
        cin>>arr[i];
    }

    int currsum[n+1];
    currsum[0] = 0;

    for(int i=1; i<=n; i++){
        currsum[i] = currsum[i-1] + arr[i-1];
    }
    int maxsum = INT_MIN;
    for(int i=1; i<=n; i++){
        int sum = 0;
        for(int j=0; j<i; j++){
            sum = currsum[i] - currsum[j];
            maxsum = max(sum, maxsum);
        }
    }
    cout<<maxsum<<endl;
    return 0;
}
```

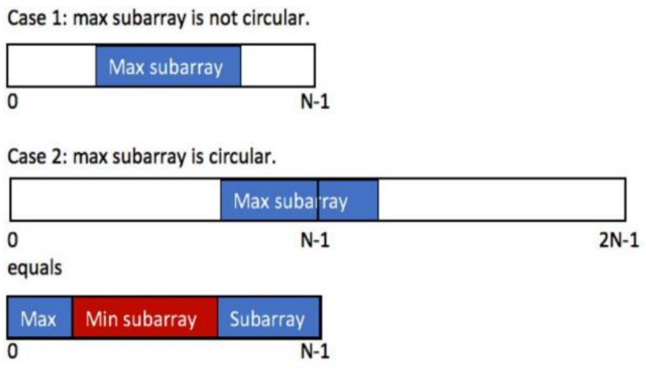
3. Kadane's Algorithm:

Idea: Start taking the sum of the array, as soon as it gets negative, discard the current subarray, and start a new sum.
Time Complexity: O(N)   Space Complexity: O(1)

```cpp
#include<bits/stdc++.h>
using namespace std;
int main()
{   int n;
    cin >> n;
    int arr[n];
    for (int i = 0; i < n; i++){
        cin >> arr[i];
    }
    int currsum = 0;
    int maxsum = INT_MIN;

    for (int i = 0; i < n; i++)
    {
        currsum += arr[i];
        if (currsum<0){
            currsum = 0;
        }
        maxsum = max(maxsum, currsum);
    }
    cout << maxsum << endl;
    return 0;
}
```

Maximum Sum Circular Subarray:

Idea: There will 2 cases,


Case 1: max subarray is not circular.
Max subarray
0
N-1
Case 2: max subarray is circular.
Max subarray
0
N-1
2N-1
equals
Max
Min subarray
Subarray
0
N-1

To get the Min subarray we multiply the array by -1 and get the maximum sum subarray.

Time Complexity: O(N)

```cpp
#include<bits/stdc++.h>
using namespace std;
/*  maximum cicular sub-array sum   */

int kadane(int arr[], int n){
    int currentsum=0;
    int maxsum=INT_MIN;
    for(int i=0; i<n; i++){
        currentsum+=arr[i];
        if(currentsum<0){
            currentsum=0;
        }
        maxsum=max(maxsum, currentsum);
    }
    return maxsum;
}
int main()
{
    int n;
    cin>>n;
    int arr[n];
    for(int i=0; i<n; i++){
        cin>>arr[i];
    }

    int wrapsum;
    int nonwrapsum;

    nonwrapsum= kadane(arr, n);
    int totalsum=0;
    for(int i=0; i<n; i++){
        totalsum+=arr[i];
        arr[i]=-arr[i];
    }

    wrapsum = totalsum + kadane(arr, n);
    cout<<max(wrapsum, nonwrapsum)<<endl;
    return 0;
}
```

Pair sum Problem:

Check if exists two element is an array such that there is equal to given **k**.

1. Time complexity: **O(n^2)**

```
#include<bits/stdc++.h>
using namespace std;
/*  Pair Sum problem Sub-array  */

bool pairsum(int arr[], int n, int k){
    for (int i=0; i<n-1; i++){
        for(int j=i+1; j<n; j++){
            if(arr[i]+arr[j]==k){
                cout<<i<<" "<<j<<endl;
                return true;
            }
        }
    }
    return false;
}
int main()
{
    int arr[]= {2,4,7,11,14,16,20,21};
    int k=31;

    cout<<pairsum(arr, 8, k)<<endl;
    return 0;
}
```

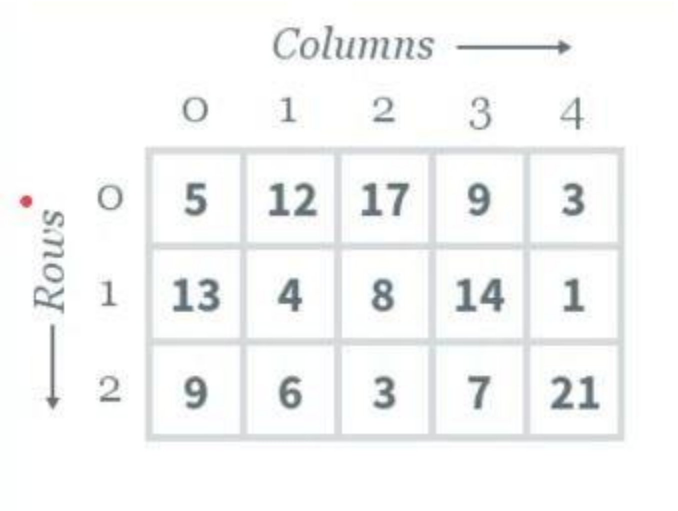
2. Time complexity: **O(n)**

```
#include<bits/stdc++.h>
using namespace std;
/*  Pair Sum problem Sub-array  in time complexity : O(n)*/

bool pairsum(int arr[], int n, int k){
    int low=0;
    int high=n-1;

    while(low<high){
        if(arr[low]+arr[high]==k){
            cout<<low<<" "<<high<<endl;
            return true;
        }
        else if(arr[low]+arr[high]>k){
            high--;
        }
        else{
            low++;
        }
    }
    return false;
}
int main()
{
    int arr[]= {2,4,7,11,14,16,20,21};
    int k=31;

    cout<<pairsum(arr, 8, k)<<endl;
    return 0;
}
```

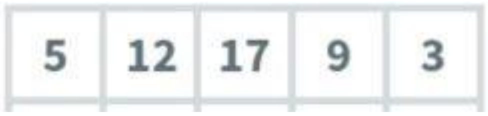
9.1 2 Dimensional Arrays

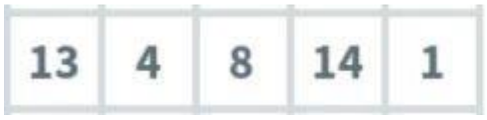
Notes

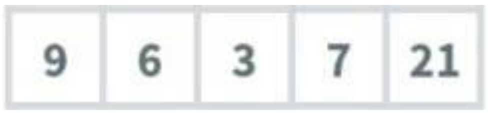
1. It is similar to 2D matrices that we studied in 11th and 12th class.



2D Array of size 3 x 5

2. It has 2 parts
   a. **Rows** - Horizontal Arrays in the 2D matrix. For instance, in the above example, we have 3 rows:
      i. 

      ii. 

      iii. 

   b. **Columns** - Vertical Arrays in the 2D matrix. For instance, in the above example, we have 5 columns:

i.

5
13
9

ii.

12
4
6

iii.

17
8
3

iv.

9
14
7

v.

3
1
21

3. Note: Indexing of both rows and columns starts with 0.

Declaration of 2D matrices

1. 2D Arrays are declared similar to 1D arrays but with an additional dimension.

   Syntax: int arr[rows][columns]

   For example:

   Int arr[n][m];

   bool arr[n][m];

   char arr[n][m];

   float arr[n][m];

Code: Taking input or print output of Array of matrx:

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    // Taking Input Array
    int n, m;
    cin>>n>>m;
    int arr[n][m];
    for(int i=0; i<n; i++){
        for(int j=0; j<m; j++){
            cin>>arr[i][j];
        }
    }
    // Printing output array
    for(int i=0; i<n; i++){
        for(int j=0; j<m; j++){
            cout<<arr[i][j]<<" ";
        }
        cout<<"\n";
    }
    return 0;
}
```

Searching in a matrix

Problem: We have to find if value x is present in the 2D array.

1. While traversing in the 2D matrix, just we have to put one if statement which checks if(a[i][j] == x) , then x is present otherwise not.

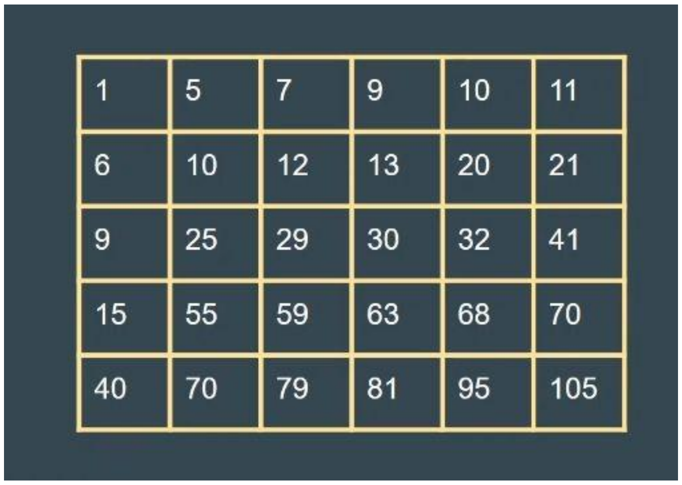
```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int n, m;
    cin>>n>>m;
    int arr[n][m];
    // Taking Input Array
    for(int i=0; i<n; i++){
        for(int j=0; j<m; j++){
            cin>>arr[i][j];
        }
    }
    int x;
    cin>>x;
    bool flag= false;
    for(int i=0; i<n; i++){
        for(int j=0; j<m; j++){
            if(arr[i][j]==x){
                cout<<i<<" "<<j<<" "<<"\n";
                flag=true;
            }
        }
    }
    if(flag){
        cout<<"Element is Found\n";
    }else{
        cout<<"Element is not found\n";
    }
    return 0;
}
```

Apni Kaksha

Spiral Order Matrix Traversal

Problem: We have to print the given 2D matrix in the spiral order. Spiral Order means that firstly, first row is printed, then last column is printed, then last row is printed and then first column is printed, then we will come inwards in the similar way.

For example:



1	5	7	9	10	11
6	10	12	13	20	21
9	25	29	30	32	41
15	55	59	63	68	70
40	70	79	81	95	105

Spiral order is given by:
1 5 7 9 10 11 21 41 70 **105 95 81 79 70** 40 15 9 6 **10 12 13** 20 32 **68 63 59** 55 25 29 30 29.

Algorithm: (We are given 2D matrix of n X m).

1. We will need 4 variables:
   a. *row_start* - initialized with **0**.
   b. *row_end* - initialized with **n-1**.
   c. *column_start* - initialized with **0**.
   d. *column_end* - initialized with **m-1**.
2. First of all, we will traverse in the row *row_start* from *column_start* to *column_end* and we will increase the *row_start* with 1 as we have traversed the starting row.
3. Then we will traverse in the column *column_end* from *row_start* to *row_end* and decrease the *column_end* by 1.
4. Then we will traverse in the row *row_end* from *column_end* to *column_start* and decrease the *row_end* by 1.
5. Then we will traverse in the column *column_start* from *row_end* to *row_start* and increase the *column_start* by 1.

6. We will do the above steps from 2 to 5 until *row_start* <= *row_end* **and** *column_start* <= *column_end*.

Code: Input the array first then perform the following code

```
#include<bits/stdc++.h>
using namespace std;
/*  spiral order matrix Print in 2D-Array   */
int main()
{
    int n,m;
    cin>>n>>m;
    int a[n][m];
    // Taking Input Array
    for(int i=0; i<n; i++){
        for(int j=0; j<m; j++){
            cin>>a[i][j];
        }
    }
    // Spiral order
    int row_start = 0, row_end = n-1, column_start = 0, column_end = m-1;
    while(row_start<=row_end && column_start<=column_end)
    {
        // for row_start
        for(int col=column_start; col<=column_end; col++){
            cout<< a[row_start][col]<<" ";
        }
        row_start++;

        // for column_end
        for(int row = row_start; row <= row_end; row++){
            cout<<a[row][column_end]<<" ";
        }
        column_end--;

        // for row_end
        for(int col=column_end; col>=column_start; col--){
            cout<<a[row_end][col]<<" ";
        }
        row_end--;

        // for column_start
        for(int row=row_end; row>=row_start; row--){
            cout<<a[row][column_start]<<" ";
        }
        column_start++;
    }
    return 0;
}
```

9.2 2D Array Challenges

Challenge 1 - Matrix Transpose

Problem

Given a square matrix A & its number of rows (or columns) N, return the transpose of A.

The transpose of a matrix is the matrix flipped over it's main diagonal, switching the row and column indices of the matrix.

Constraints

1 <= N <= 1000

Sample Input1

```
A = [
        [1,2,3],
        [4,5,6],
        [7,8,9]
    ]
N = 3
```

Sample Output1

```
A = [
        [1,4,7],
        [2,5,8],
        [3,6,9]
]
```

Approach

Transpose of a matrix means swapping its rows with columns & columns with rows. But this swap is to be done only for the upper triangle of a matrix i.e. swap half of the elements of the diagonally upper half of the matrix withthe diagonally lower half once. In this, each (row, col) & (col, row) pair will be swapped exactly once and the transpose of the square matrix could be obtained.

Code

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

int main()
{
    int N = 3;
    int A[N][N] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};

    for(int i=0; i<N; i++){
        for(int j=i; j<N; j++){
            //swap
            int temp = A[i][j];
            A[i][j] = A[j][i];
            A[j][i] = temp;
        }
    }
    //print transpose for(int i=0; i<N; i++) {
    for(int i=0; i<N; i++){
        for(int j=0; j<N; j++){
            cout<<A[i][j]<<" ";
        }
        cout<<"\n";
    }
    return 0;
}
```

Time complexity : O(N^2)

Space complexity : O(1), as no extra space for a new matrix was used

Apni Kaksha

Challenge 2 - Matrix Multiplication

Problem

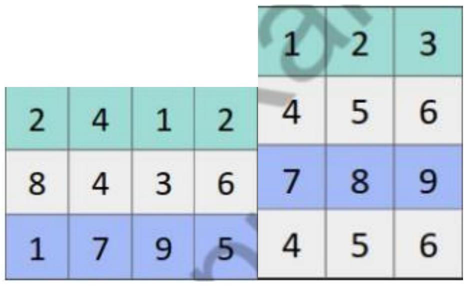
Given two 2-Dimensional arrays of sizes n1 x n2 and n2 x n3. Your task is to multiply these matrices and output the multiplied matrix.

Constraints

1 <= n1,n2,n3 <= 1,000

Sample test case:

Input

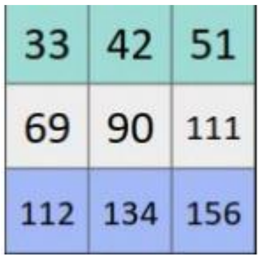


2	4	1	2
8	4	3	6
1	7	9	5

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

3x4 **4x3**

Output



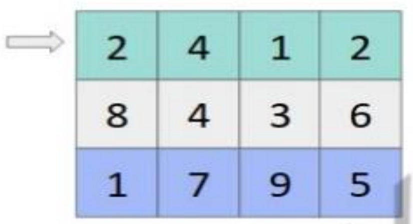
33	42	51
69	90	111
112	134	156

Approach

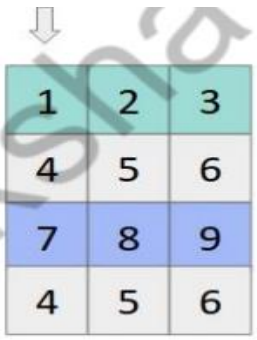
1. Make a nested loop of order 3. In the outer loop iterate over rows of the first matrix and in the inner loop iterate over columns of the second matrix.
2. Multiply rows of the first matrix with columns of the second matrix inthe innermost loop and update in the answer matrix

Dry Run

First Iteration

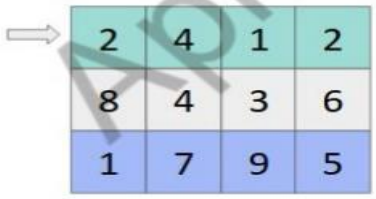


2	4	1	2
8	4	3	6
1	7	9	5

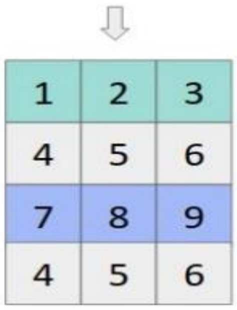


1	2	3
4	5	6
7	8	9
4	5	6

Second Iteration

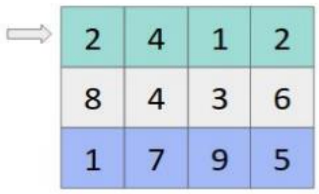


2	4	1	2
8	4	3	6
1	7	9	5

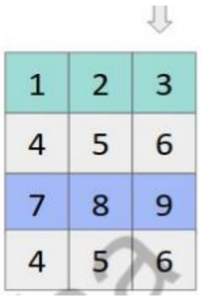


1	2	3
4	5	6
7	8	9
4	5	6

Third Iteration

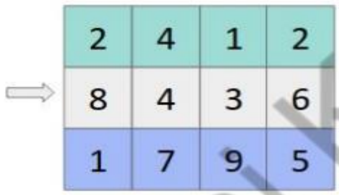


2	4	1	2
8	4	3	6
1	7	9	5

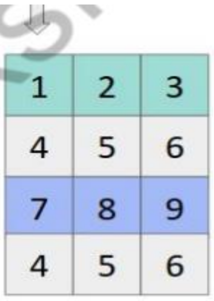


1	2	3
4	5	6
7	8	9
4	5	6

Fourth Iteration

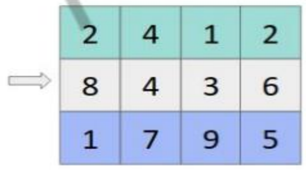


2	4	1	2
8	4	3	6
1	7	9	5

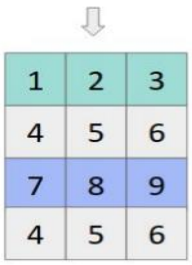


1	2	3
4	5	6
7	8	9
4	5	6

Fifth Iteration

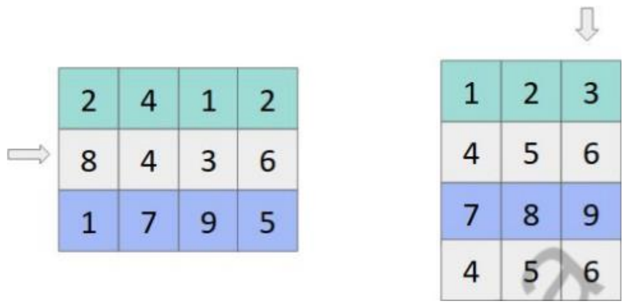


2	4	1	2
8	4	3	6
1	7	9	5

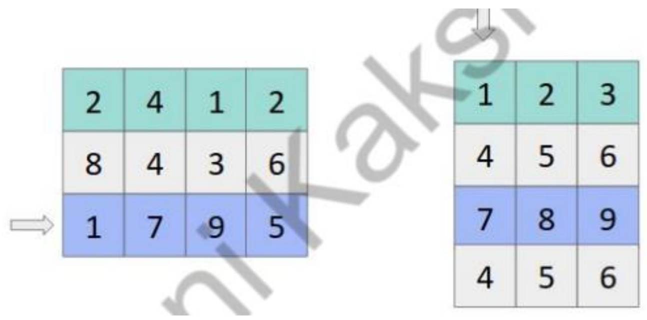


1	2	3
4	5	6
7	8	9
4	5	6

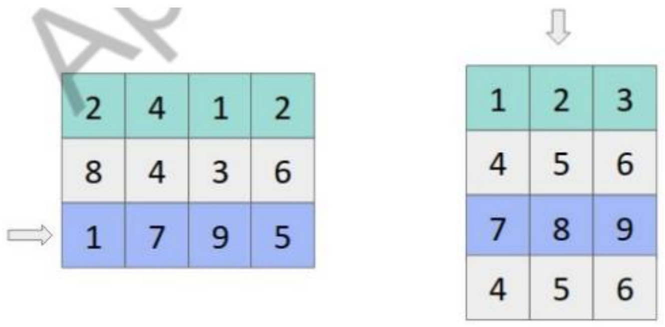
Sixth Iteration



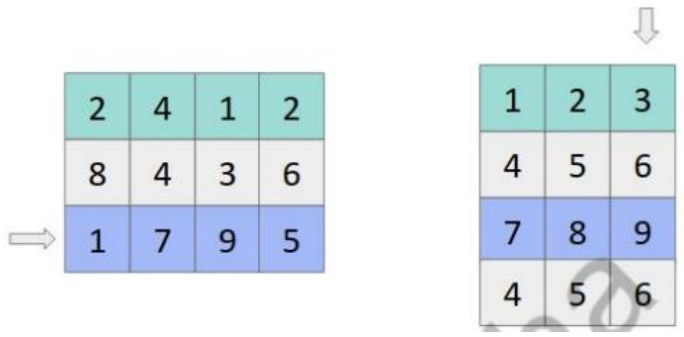
Seventh Iteration



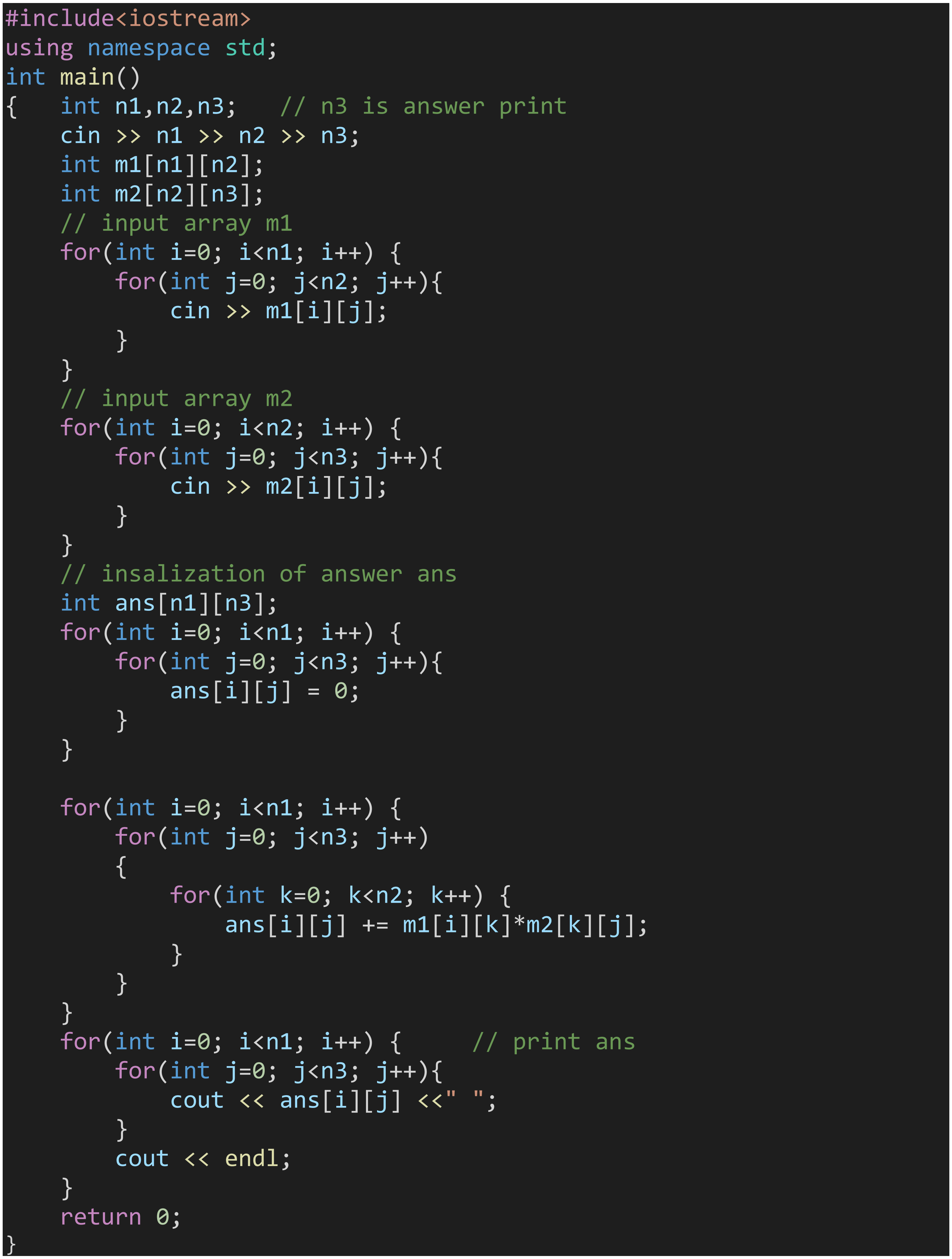
Eighth Iteration



Ninth Iteration



Time Complexity: O(n1*n2*n3)



```cpp
#include<iostream>
using namespace std;
int main()
{   int n1,n2,n3;   // n3 is answer print
    cin >> n1 >> n2 >> n3;
    int m1[n1][n2];
    int m2[n2][n3];
    // input array m1
    for(int i=0; i<n1; i++) {
        for(int j=0; j<n2; j++){
            cin >> m1[i][j];
        }
    }
    // input array m2
    for(int i=0; i<n2; i++) {
        for(int j=0; j<n3; j++){
            cin >> m2[i][j];
        }
    }
    // insalization of answer ans
    int ans[n1][n3];
    for(int i=0; i<n1; i++) {
        for(int j=0; j<n3; j++){
            ans[i][j] = 0;
        }
    }

    for(int i=0; i<n1; i++) {
        for(int j=0; j<n3; j++)
        {
            for(int k=0; k<n2; k++) {
                ans[i][j] += m1[i][k]*m2[k][j];
            }
        }
    }
    for(int i=0; i<n1; i++) {      // print ans
        for(int j=0; j<n3; j++){
            cout << ans[i][j] <<" ";
        }
        cout << endl;
    }
    return 0;
}
```

Challenge 3 - 2D matrix Search

Problem

Given **n x m** matrix. Write an algorithm to find that the given value exists in the matrix or not.Integers in each row are sorted in ascending from left to right. Integers in each column are sorted in ascending from top to bottom.

Constraints

1 <= N,M <= 1,000

Sample Test Case:

Consider the following matrix:

[1, 4, 7, 11, 15],

[2, 5, 8, 12, 19],

[3, 6, 9, 16, 22],

[10, 13, 14, 17, 24],

[18, 21, 23, 26, 30]

Given target = 5, return true.
Given target = 20, return false.

Solution 1 : <u>Brute Force Approach</u>   **Q.** Linear search in a 2D Array.

```
#include "bits/stdc++.h"
using namespace std;
int main()
{   int n, m;
    cin >> n >> m;
    int target;
    cin >> target;
    int mat[n][m];
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            cin >> mat[i][j];
        }
    }
    bool found = false;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            if (mat[i][j] == target){
                found = true;
            }
        }
    }
    if (found){
    cout << "Found";
    }
    else{
    cout << "Not Found";
    }return 0;
}
```

Time complexity : O(N*M)

Solution 2: Optimised Approach [IMP]

1. Start from the top right element.
2. You are at (r,c)

```
if(matrix[r][c] == target)
        return true
If (matrix[r][c] > target)c-
        -
else
        r++;
```

At (r,c), you can go to (r-1,c) or (r,c-1), depending on the value of matrix[i][j] and target.

```cpp
#include "bits/stdc++.h"
using namespace std;
int main() {
    int n, m;
    cin >> n >> m;
    int target;
    cin >> target;
    int mat[n][m];
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            cin >> mat[i][j];
        }
    }
    bool found = false;
    int r = 0, c = n - 1;
    while (r < m && c >= 0) {
        if (mat[r][c] == target) {
            found = true;
        }
        mat[r][c] > target ? c-- : r++;
    }
    if (found){
        cout << "Found";
    }else{
    cout << "Not Found";
    }
}
```

Time complexity : O(N + M)

10. Character Arrays (char)

Arrays of characters are known as Character Arrays.

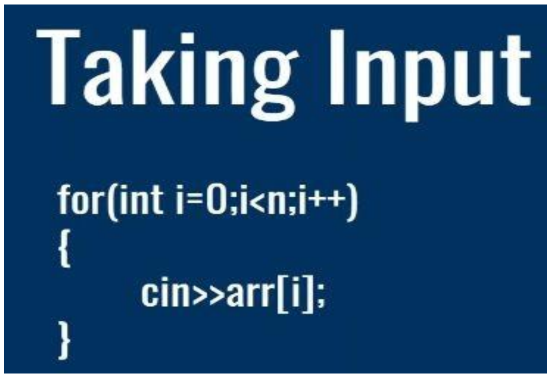
Declaration

To declare a character array of n size, we do

char arr[n+1];

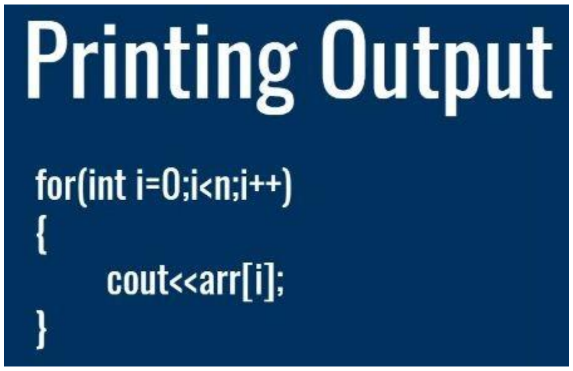
Note: We declare an array of n+1 as 0 to n-1 indices store the actual string and n^th character stores '\0' (null character).

Taking input and **Printing output**



Taking Input

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



Printing Output

```
for(int i=0;i<n;i++)
{
        cout<<arr[i];
}
```

We can also directly take input if there are no spaces between the characters in the word

cin >> arr;

In the similar way, Print character.

cout << arr;

Important Questions

1. **Check if a given character array is a palindrome or not.**

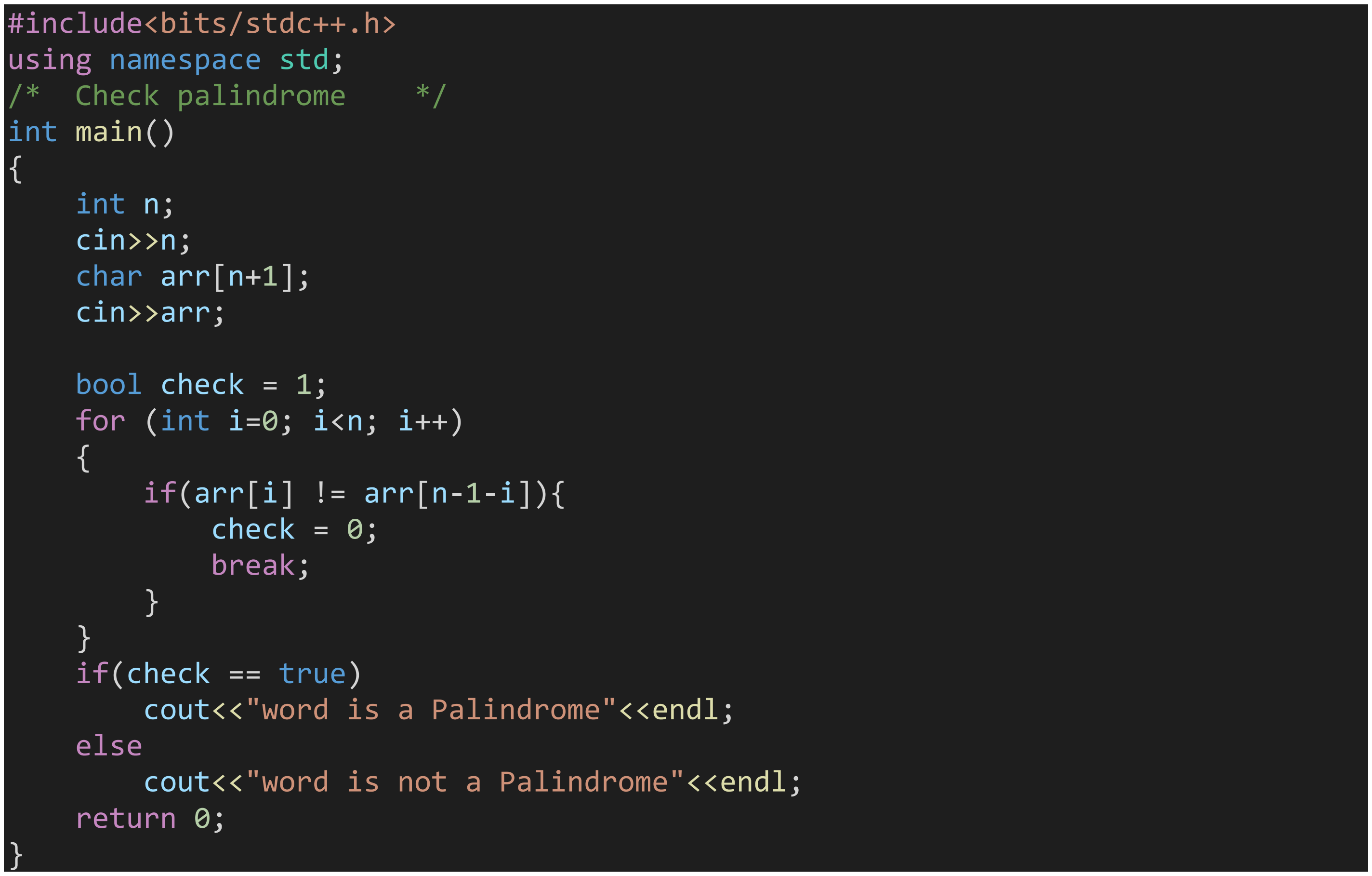
Palindrome: Given a string s, on reversing the string we get the same string we call that string is a palindrome.

For example:

raceca

Algorithm:

1) Let the length of the character array be n.
2) Keep a boolean variable ans to store the result and initialize it with true.
2) Iterate over the string and check if i^th character is equal to (n-i-1)^th, there can be2 cases
    a) If equal, then do nothing
    b) If unequal, then put ans = false
3) When the loop ends, if ans is true, then the string is palindrome else it is not a palindrome.



```
#include<bits/stdc++.h>
using namespace std;
/*  Check palindrome    */
int main()
{
    int n;
    cin>>n;
    char arr[n+1];
    cin>>arr;

    bool check = 1;
    for (int i=0; i<n; i++)
    {
        if(arr[i] != arr[n-1-i]){
            check = 0;
            break;
        }
    }
    if(check == true)
        cout<<"word is a Palindrome"<<endl;
    else
        cout<<"word is not a Palindrome"<<endl;
    return 0;
}
```

2. Question: Largest word in a sentence

To input a complete sentence, we use the getline() function.

cin.getline(arr, n);

where arr is the character array and n is the total length of sentence

Approach

1. Iterate over the sentence and keep variables currLen and maxLen which store the current length of the present word being iterated and the overall maximum length word's length.

2. Whenever we encounter a space during iteration, we will maximize our maxLen variable.

maxLen = max(maxLen, currLen)

```cpp
#include<bits/stdc++.h>
using namespace std;
/*  Large Worde in Sentance of char array   */
int main()
{   int n;
    cin>>n;
    cin.ignore();
    char arr[n+1];

    cin.getline(arr, n); //in_build function used
    cin.ignore();

    int i=0;
    int currLen = 0, maxLen = 0;
    int st=0, maxst=0;
    while(1)
    {
        if(arr[i] == ' ' || arr[i] == '\0')      // ' ' space used
        {
            if(currLen > maxLen){
                maxLen = currLen;
                maxst = st;
            }
            currLen = 0;
            st = i+1;
        }
        else
        currLen++;
        if(arr[i] == '\0')
            break;
        i++;
    }
    cout<<maxLen<<endl;
    for(int i=0; i<maxLen; i++){
        cout<<arr[i+maxst];
    }
    return 0;
}
```