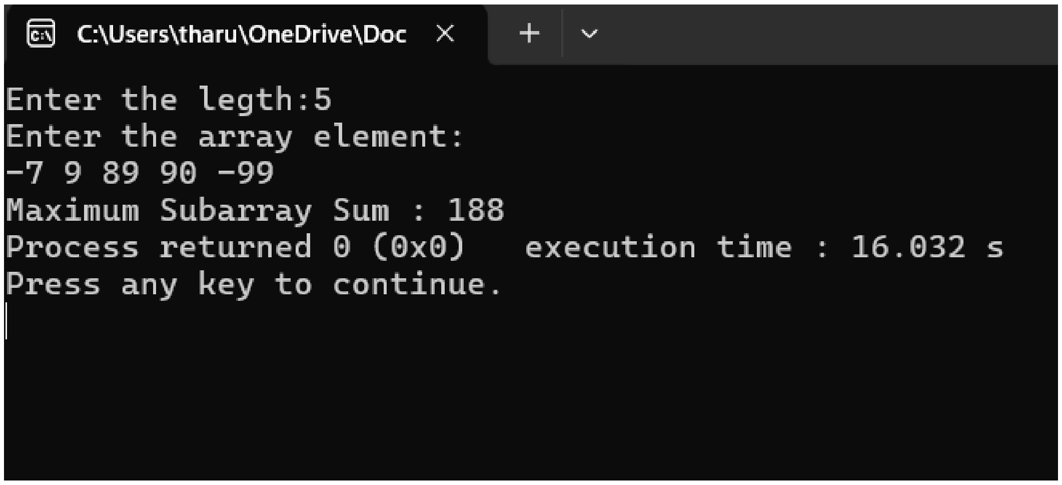
1. Maximum Subarray Sum – Kadane's Algorithm:

Given an array arr[], the task is to find the subarray that has the maximum sum and return its sum.

```
#include <iostream>
using namespace std;
int main(){
   int len;
   cout<<"Enter the legth:";
   cin>>len;
   int arr[len];
   cout<<"Enter the array element:"<<endl;
   for(int i=0;i<len;i++){
      cin>>arr[i];
   }
   //Kadane's Algorithm
   long long maxsum=LONG_MIN;
   long long sum=0;
   for(auto i:arr){
      sum=max(sum,sum+i);
      maxsum=max(maxsum,sum);
   }
   cout<<"Maximum Subarray Sum : "<<maxsum;
   return 0;
}
```

Output :



```
C:\Users\tharu\OneDrive\Doc
Enter the legth:5
Enter the array element:
-7 9 89 90 -99
Maximum Subarray Sum : 188
Process returned 0 (0x0)   execution time : 16.032 s
Press any key to continue.
```

Time complexity : O(n)

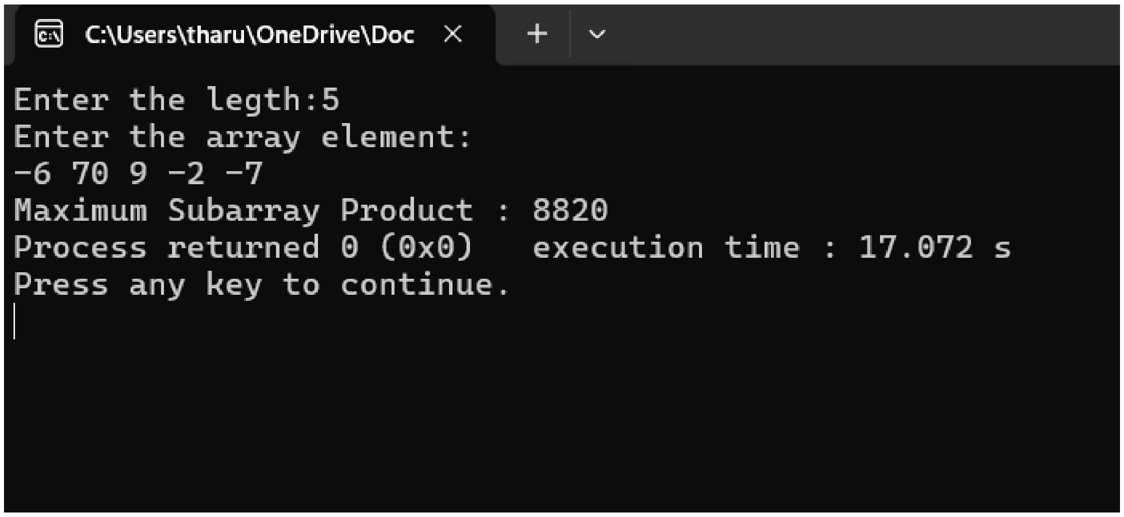
Space complexity : O(1)

2.Maximum Product Subarray

Given an integer array, the task is to find the maximum product of any subarray.

```
#include <iostream>
#include <math.h>
using namespace std;
int main(){
  int len;
  cout<<"Enter the legth:";
  cin>>len;
  long long arr[len];
  cout<<"Enter the array element:"<<endl;
  for(int i=0;i<len;i++){
    cin>>arr[i];
  }
  //Maximum Product Algorithm
  long long maxproduct=arr[0];
  long long minval=arr[0];
  long long maxval=arr[0];
  for(int i=1;i<len;i++){
    if(arr[i]<0) swap(minval,maxval);
    minval=min(arr[i],minval*arr[i]);
    maxval=max(arr[i],maxval*arr[i]);
    maxproduct=max(maxproduct,maxval);
  }
  cout<<"Maximum Subarray Product : "<<maxproduct;
  return 0;
}
```

Output:



```
Enter the legth:5
Enter the array element:
-6 70 9 -2 -7
Maximum Subarray Product : 8820
Process returned 0 (0x0)   execution time : 17.072 s
Press any key to continue.
```

Time complexity : O(n)

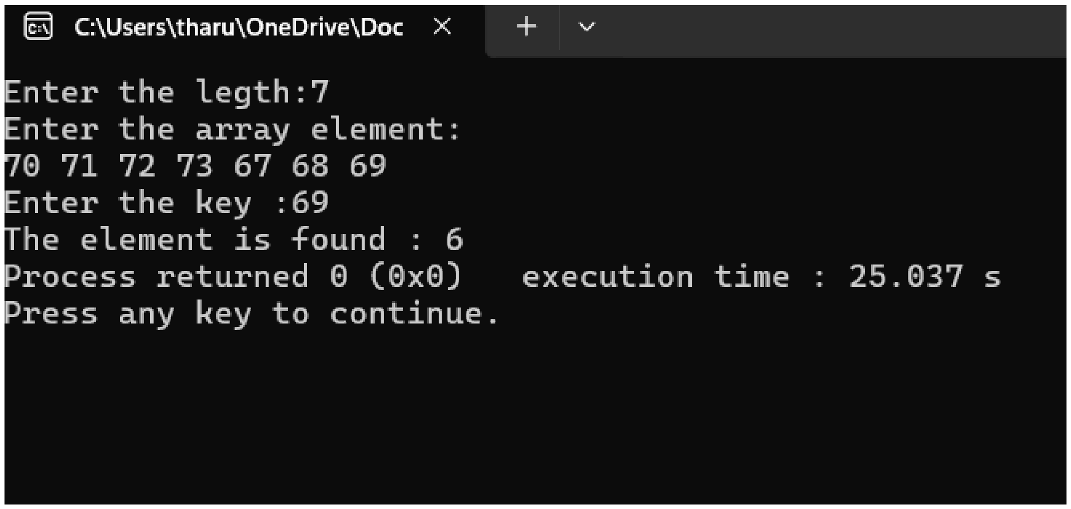
Space complexity : O(1)

3.Search in a sorted and rotated Array

Given a sorted and rotated array arr[] of n distinct elements, the task is to find the index of given key in the array. If the key is not present in the array, return -1.

```
#include <iostream>
using namespace std;
int binary_search(int left,int right,long long arr[],int target){
  while(left<right){
    int mid=(left+right)/2;
    if(arr[mid]==target) return mid;
    else if(arr[mid]<target){
      left=mid+1;
    }
    else{
      right=mid-1;
    }
  }
  return -1;
}
int peak_element(int left,int right,long long arr[],int target){
  while(left<right){
    int mid=(left+right)/2;
    if(arr[mid+1]<arr[mid] && arr[mid]<arr[mid-1]){
      right=mid;
    }
    else{
      left=mid+1;
    }
  }
  return left;
}
int main(){
  int len;
  cout<<"Enter the legth:";
  cin>>len;
  long long arr[len];
  cout<<"Enter the array element:"<<endl;
  for(int i=0;i<len;i++){
    cin>>arr[i];
  }
  int target;
  cout<<"Enter the key :";
  cin>>target;
  int idx=peak_element(0,len-1,arr,target);
  int idx1=binary_search(0,idx-1,arr,target);
  int idx2=binary_search(idx+1,len-1,arr,target);
  if(arr[idx]==target){
    cout<<"The element is found : "<<idx;
  }
else{
    if(idx1==-1 && idx2==-1) cout<<"The element is NotFound : "<<idx1;
    else if(idx1==-1) cout<<"The element is found : "<<idx2;
    else cout<<"The element is found : "<<idx1;
}
return 0;

}
```

Output:



C:\Users\tharu\OneDrive\Doc

```
Enter the legth:7
Enter the array element:
70 71 72 73 67 68 69
Enter the key :69
The element is found : 6
Process returned 0 (0x0)   execution time : 25.037 s
Press any key to continue.
```

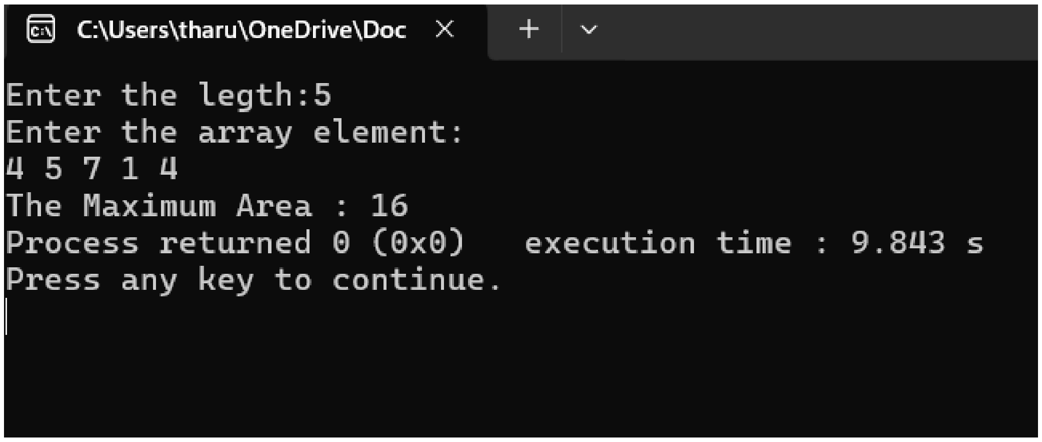
Time complexity : O(log(n))

Space complexity : O(1)

4.Container with Most Water

```
#include <iostream>
using namespace std;
int main(){
    int len;
    cout<<"Enter the legth:";
    cin>>len;
    long long arr[len];
    cout<<"Enter the array element:"<<endl;
    for(int i=0;i<len;i++){
        cin>>arr[i];
    }
    int left=0;
    int right=len-1;
    long area=0;
    while(left<right){
        int b=min(arr[left],arr[right]);
        int l=(right-left);
        area=max(area,(long)l*b);
        if(arr[left]<arr[right]){
            left++;
        }
        else{
            right--;
        }
    }
    cout<<"The Maximum Area : "<<area;
    return 0;
}
```

Output:



```
Enter the legth:5
Enter the array element:
4 5 7 1 4
The Maximum Area : 16
Process returned 0 (0x0)   execution time : 9.843 s
Press any key to continue.
```

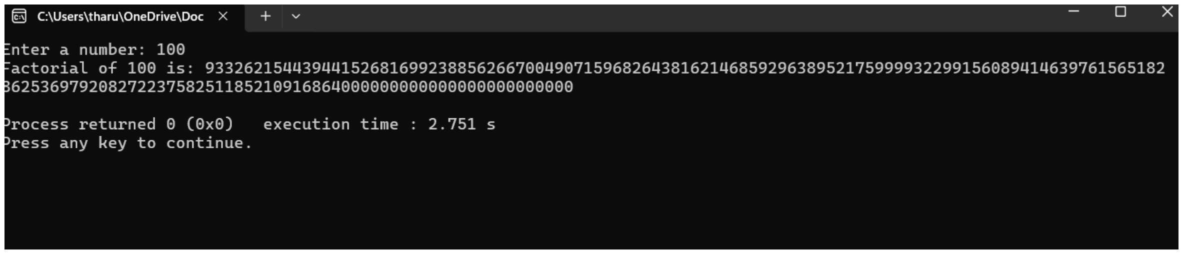
Time complexity : O(n)

Space complexity : O(1)

5.Find the Factorial of a large number

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

void multiply(vector<int>& result, int num) {
    int carry = 0;
    for (int i = 0; i < result.size(); i++) {
        int prod = result[i] * num + carry;
        result[i] = prod % 10;
        carry = prod / 10;
    }
    while (carry) {
        result.push_back(carry % 10);
        carry /= 10;
    }
}

void factorial(int n) {
    vector<int> result;
    result.push_back(1);

    for (int i = 2; i <= n; i++) {
        multiply(result, i);
    }
    for (int i = result.size() - 1; i >= 0; i--) {
        cout << result[i];
    }
    cout << endl;
}

int main() {
    int n;
    cout << "Enter a number: ";
    cin >> n;
    cout << "Factorial of " << n << " is: ";
    factorial(n);
    return 0;
}
```

Output:



C:\Users\tharu\OneDrive\Doc

```
Enter a number: 100
Factorial of 100 is: 933262154439441526816992388562667004907159682643816214685929638952175999932299156089414639761565182
8625369792082722375825118521091686400000000000000000000000000

Process returned 0 (0x0)   execution time : 2.751 s
Press any key to continue.
```

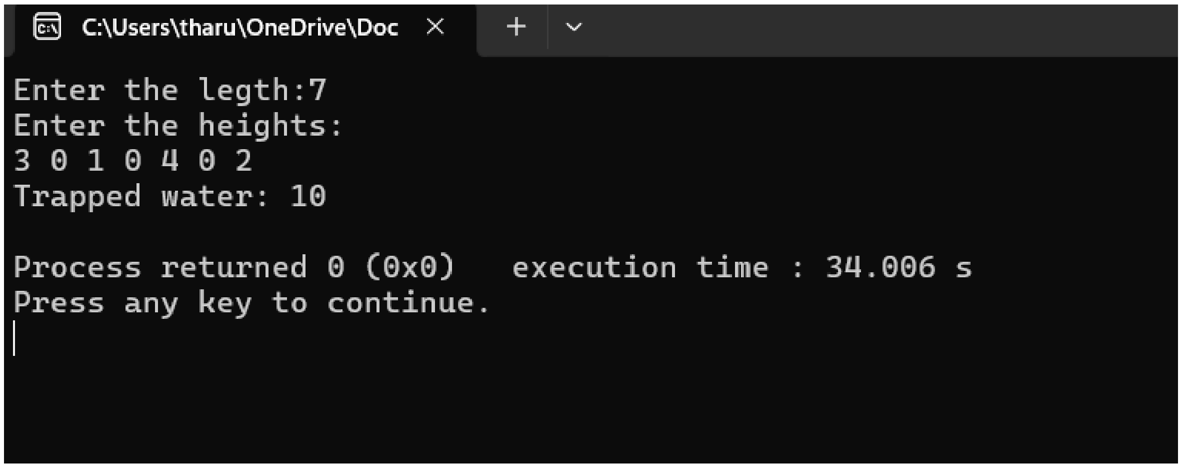
Time complexity : O(n*d);

Space Complexity : O(d)

6.Trapping Rainwater Problem states that given an array of n non-negative integers arr[] representing an elevation map where the width of each bar is 1, compute how much water it can trap after rain.

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

int trap(vector<int>& height) {
   int waterAmount = 0;
   int l = 0, r = height.size() - 1;
   int lmax = height[l], rmax = height[r];

   while (l < r) {
      if (lmax < rmax) {
         l++;
         lmax = max(lmax, height[l]);
         waterAmount += (lmax - height[l]);
      } else {
         r--;
         rmax = max(rmax, height[r]);
         waterAmount += (rmax - height[r]);
      }
   }

   return waterAmount;
}

int main() {
   int len;
   cout<<"Enter the legth:";
   cin>>len;
   vector<int> arr(len);
   cout<<"Enter the heights:"<<endl;
   for(int i=0;i<len;i++){
      cin>>arr[i];
   }
   cout << "Trapped water: " << trap(arr) << endl;
   return 0;
}
```

Output:



```
C:\Users\tharu\OneDrive\Doc

Enter the legth:7
Enter the heights:
3 0 1 0 4 0 2
Trapped water: 10

Process returned 0 (0x0)   execution time : 34.006 s
Press any key to continue.
```

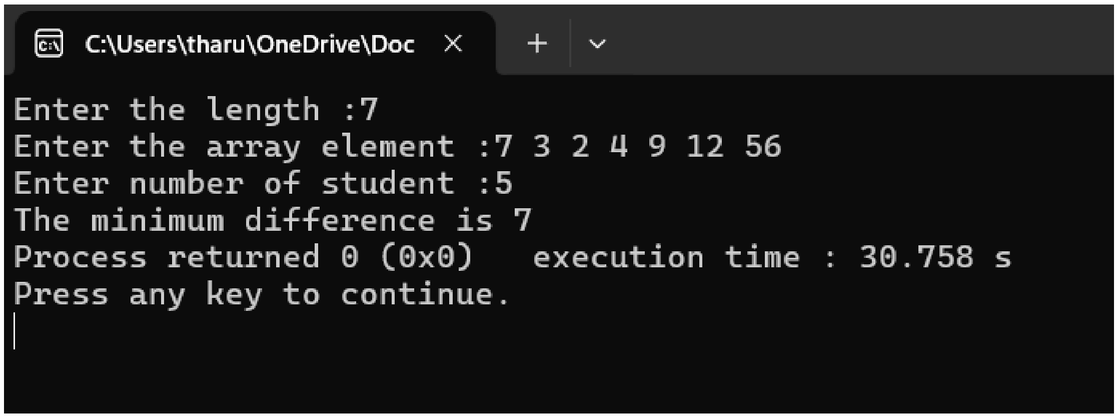
Time Complexity : O(n);

Space Complexity : O(1);

7.Chocolate Distribution Problem Given an array arr[] of n integers where arr[i] represents the number of chocolates in ith packet. Each packet can have a variable number of chocolates. There are m students, the task is to distribute chocolate packets such that Each student gets exactly one packet. The difference between the maximum and minimum number of chocolates in the packets given to the students is minimized.

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <limits.h>
using namespace std;
int main(){
  int len,m;
  cout<<"Enter the length :";
  cin>>len;
  vector<int> arr(len);
  cout<<"Enter the array element :";
  for(int i=0;i<len;i++){
    cin>>arr[i];
  }
  cout<<"Enter number of student :";
  cin>>m;
  sort(arr.begin(),arr.end());
  int maxval=INT_MAX;
  for(int i=0;i<len-m;i++){
    int val=arr[i+m-1]-arr[i];
    maxval=min(maxval,val);
  }
  cout<<"The minimum difference is "<<maxval;
  return 0;
 }
```

Output:



```
Enter the length :7
Enter the array element :7 3 2 4 9 12 56
Enter number of student :5
The minimum difference is 7
Process returned 0 (0x0)   execution time : 30.758 s
Press any key to continue.
```

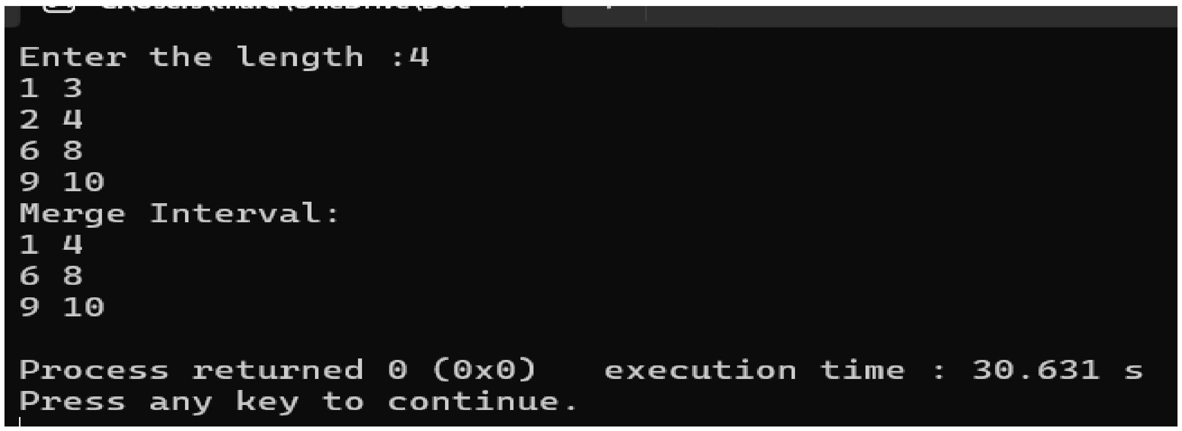
Time Complexity : O(nlog(n))

Space Complexity : O(1)

8.Merge Overlapping Interval Given an array of time intervals where arr[i] = [starti, endi], the task is to merge all the overlapping intervals into one and output the result which should have only mutually exclusive intervals.

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
int main(){
   int len;
   cout<<"Enter the length :";
   cin>>len;
   vector<vector<int>> arr(len);
   for(int i=0;i<len;i++){
      int a,b;
      cin>>a>>b;
      arr[i].push_back(a);
      arr[i].push_back(b);
   }
   sort(arr.begin(),arr.end());
   vector<vector<int>> res;
   res.push_back(arr[0]);
   for(int i=1;i<len;i++){
      int n=res.size()-1;
      if(res[n][1]>=arr[i][0]){
         res[n][1]=max(res[n][1],arr[i][1]);
      }
      else{
         res.push_back(arr[i]);
      }
   }
   cout<<"Merge Interval:"<<endl;
   for(auto i:res){
      cout<<i[0]<<" "<<i[1]<<endl;
   }
   return 0;
}
```

Output:



```
Enter the length :4
1 3
2 4
6 8
9 10
Merge Interval:
1 4
6 8
9 10

Process returned 0 (0x0)   execution time : 30.631 s
Press any key to continue.
```

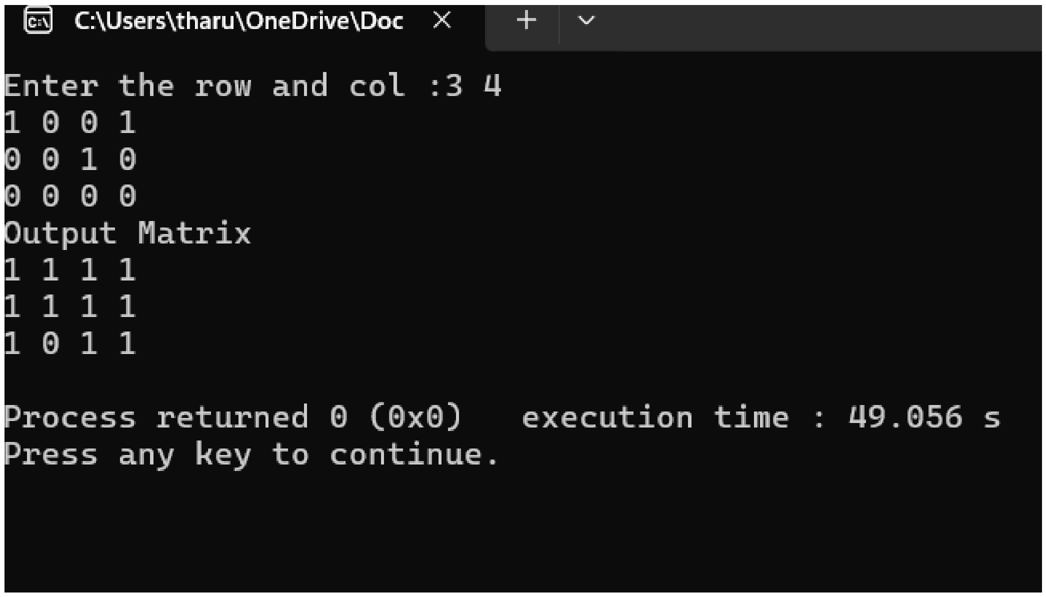
Time Complexity : O(N log(N))

Space Complexity : O(N)

A Boolean Matrix Question Given a boolean matrix mat[M][N] of size M X N, modify it such that if a matrix cell mat[i][j] is 1 (or true) then make all the cells of ith row and jth column as 1.

```
#include <iostream>
#include <vector>
using namespace std;
int main(){
   int m,n;
   cout<<"Enter the row and col :";
   cin>>m>>n;
   vector<vector<int>> arr(m,vector<int>(n,0));
   for(int i=0;i<m;i++){
      for(int j=0;j<n;j++){
         cin>>arr[i][j];
      }
   }
   vector<int> row(m);
   vector<int> col(n);
   for(int i=0;i<m;i++){
      for(int j=0;j<n;j++){
         if(arr[i][j]==1){
            row[i]=1;
            col[j]=1;
         }
      }
   }
   for(int i=0;i<m;i++){
      for(int j=0;j<n;j++){
         if(col[j]==1 || row[i]==1){
            arr[i][j]=1;
         }
      }
   }
   cout<<"Output Matrix "<<endl;
   for(int i=0;i<m;i++){
      for(int j=0;j<n;j++){
         cout<<arr[i][j]<<" ";
      }
      cout<<endl;
   }
   return 0;
}
```

Output:



```
Enter the row and col :3 4
1 0 0 1
0 0 1 0
0 0 0 0
Output Matrix
1 1 1 1
1 1 1 1
1 0 1 1

Process returned 0 (0x0)   execution time : 49.056 s
Press any key to continue.
```

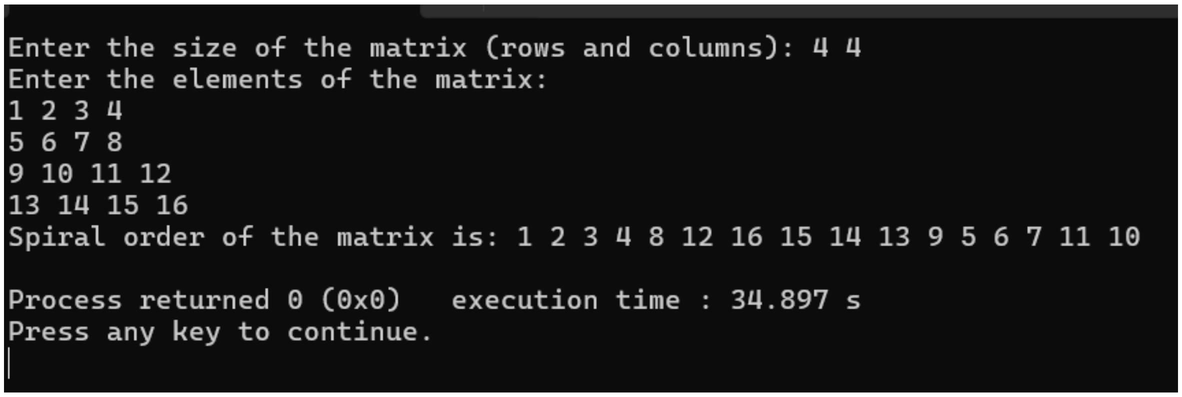
Time Complexity : O(M*N)

Space Complexity : O(M+N)

10.Print a given matrix in spiral for a Given an m x n matrix, the task is to print all elements of the matrix in spiral form.

```
#include <iostream>
#include <vector>
using namespace std;
vector<int> spiralOrder(vector<vector<int>>& matrix) {
  vector<int> ans;
  int row = 0, rowls = matrix.size() - 1;
  int col = 0, colls = matrix[0].size() - 1;

  while (row <= rowls && col <= colls) {
    for (int i = col; i <= colls; i++) {
      ans.push_back(matrix[row][i]);
    }
    row++;
    for (int i = row; i <= rowls; i++) {
      ans.push_back(matrix[i][colls]);
    }
    colls--;
    if (row <= rowls) {
      for (int i = colls; i >= col; i--) {
        ans.push_back(matrix[rowls][i]);
      }
      rowls--;
    }
    if (col <= colls) {
      for (int i = rowls; i >= row; i--) {
        ans.push_back(matrix[i][col]);
      }
      col++;
    }
  }
  return ans;
}
int main() {
  int rows, cols;
  cout << "Enter the size of the matrix (rows and columns): ";
  cin >> rows >> cols;
  vector<vector<int>> matrix(rows, vector<int>(cols));
  cout << "Enter the elements of the matrix:" << endl;
  for (int i = 0; i < rows; i++) {
    for (int j = 0; j < cols; j++) {
      cin >> matrix[i][j];
    }
  }
  vector<int> result = spiralOrder(matrix);
  cout << "Spiral order of the matrix is: ";
  for (int i = 0; i < result.size(); i++) {
    cout << result[i] << " ";
  }
  cout << endl;
  return 0;
}
```

Output:



```
Enter the size of the matrix (rows and columns): 4 4
Enter the elements of the matrix:
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
Spiral order of the matrix is: 1 2 3 4 8 12 16 15 14 13 9 5 6 7 11 10

Process returned 0 (0x0)   execution time : 34.897 s
Press any key to continue.
```

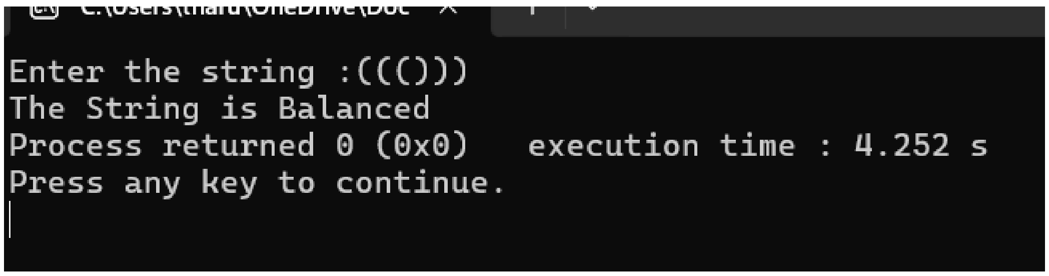
Time Complexity : O(M*N)

Space Complexity : O(M*N)

13.Check if given Parentheses expression is balanced or not.

```
#include <iostream>
#include <stack>
using namespace std;
int main(){
   string s;
   cout<<"Enter the string :";
   cin >> s;
   stack<char> stack;
   for(auto i : s){
      if(!stack.empty() && stack.top() == '(' && i == ')') stack.pop();
      else stack.push(i);
   }
   if(stack.empty()) cout << "The String is Balanced";
   else cout << "The String is not Balanced";
   return 0;
}
```

Output:



```
Enter the string :((()))
The String is Balanced
Process returned 0 (0x0)   execution time : 4.252 s
Press any key to continue.
```

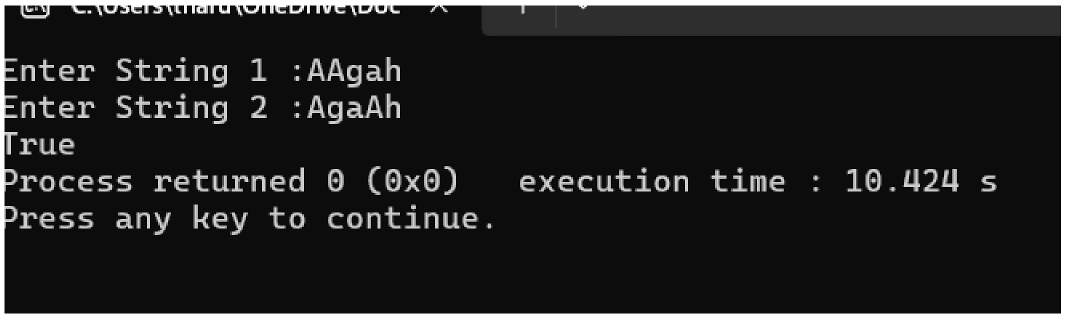
Time Complexity : O(N)

Space Complexity : O(N)

14.Check if two Strings are Anagrams of each other Given two strings s1 and s2 consisting of lowercase characters, the task is to check whether the two given strings are anagrams of each other or not. An anagram of a string is another string that contains the same characters, only the order of characters can be different.

```
#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;
int main(){
   string s1,s2;
   cout<<"Enter String 1 :";
   cin>>s1;
   cout<<"Enter String 2 :";
   cin>>s2;
   vector<int> arr(52,0);
   for(auto i:s1){
      if(i>='a' && i<='z') arr[i-'a']++;
      else if(i>='A' && i<='Z') arr[i-'A'+26]++;
   }
   for(auto i:s2){
      if(i>='a' && i<='z') arr[i-'a']--;
      else if(i>='A' && i<='Z') arr[i-'A'+26]--;
   }
   int flag=1;
   for(auto i:arr){
      if(i!=0){
         flag=0;
         break;
      }
   }
   if(flag) cout<<"True";
   else cout<<"False";
   return 0;
}
```

Output:



```
Enter String 1 :AAgah
Enter String 2 :AgaAh
True
Process returned 0 (0x0)   execution time : 10.424 s
Press any key to continue.
```

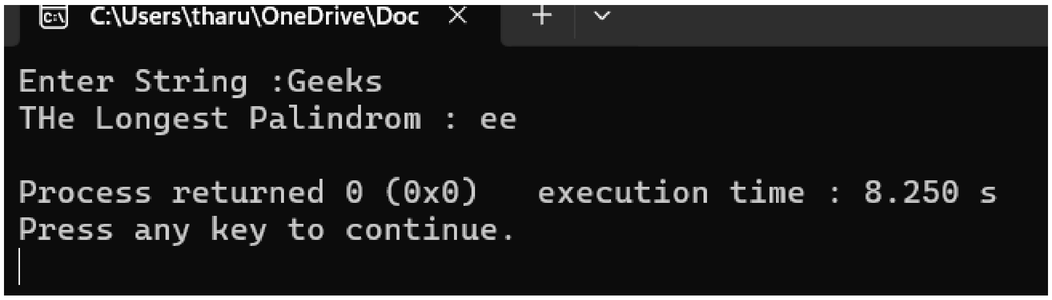
Time Complexity : O(M+N)

Space Complexity : O(1)

15.Longest Palindromic Substring Given a string str, the task is to find the longest substring which is a palindrome. If there are multiple answers, then return the first appearing substring.

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

int expandAroundCenter(string s, int left, int right) {
   while (left >= 0 && right < s.length() && s[left] == s[right]) {
      left--;
      right++;
   }
   return right - left - 1;
}

string longestPalindrome(string s) {
   if (s.empty()) {
      return "";
   }

   int start = 0, end = 0;

   for (int i = 0; i < s.length(); i++) {
      int odd = expandAroundCenter(s, i, i);
      int even = expandAroundCenter(s, i, i + 1);
      int max_len = max(odd, even);

      if (max_len > end - start) {
         start = i - (max_len - 1) / 2;
         end = i + max_len / 2;
      }
   }

   return s.substr(start, end - start + 1);
}

int main() {
   string s;
   cout<<"Enter String :";
   cin >> s;
   cout <<"THe Longest Palindrom : "<<longestPalindrome(s) << endl;
   return 0;
}
```

Output:



```
C:\Users\tharu\OneDrive\Doc

Enter String :Geeks
THe Longest Palindrom : ee

Process returned 0 (0x0)   execution time : 8.250 s
Press any key to continue.
```

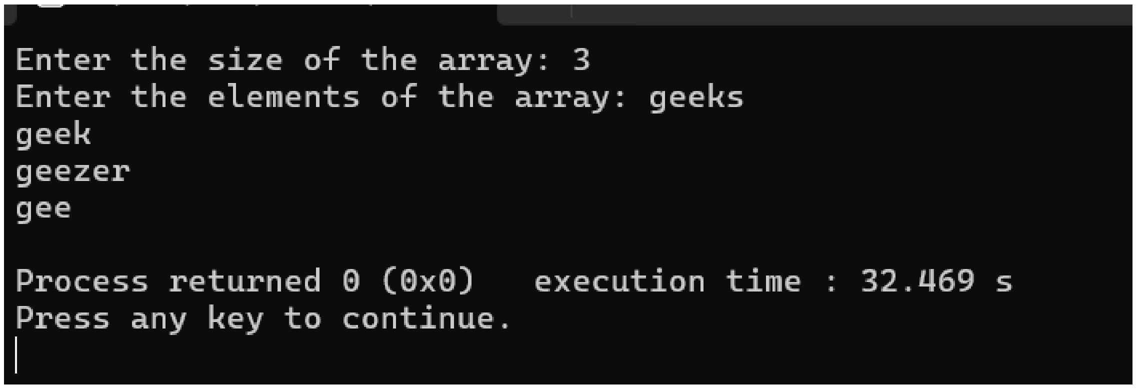
Time Complexity : O(N)

Space Complexity : O(1)

16.Longest Common Prefix using Sorting Given an array of strings arr[]. The task is to return the longest common prefix among each and every strings present in the array. If there‟s no prefix common in all the strings, return “-1”.

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

string longestCommonPrefix(vector<string>& strs) {
   if (strs.empty()) return "";
   string prefix = strs[0];
   for (int i = 1; i < strs.size(); i++) {
      while (strs[i].find(prefix) != 0) {
         prefix = prefix.substr(0, prefix.length() - 1);
         if (prefix.empty()) return "";
      }
   }
   return prefix;
}

int main() {
   int n;
   cout << "Enter the size of the array: ";
   cin >> n;
   cin.ignore();
   vector<string> nums(n);
   cout << "Enter the elements of the array: ";
   for (int i = 0; i < n; i++) {
      getline(cin, nums[i]);
   }
   cout << longestCommonPrefix(nums) << endl;
   return 0;
}
```

Ouutput:



```
Enter the size of the array: 3
Enter the elements of the array: geeks
geek
geezer
gee

Process returned 0 (0x0)   execution time : 32.469 s
Press any key to continue.
```

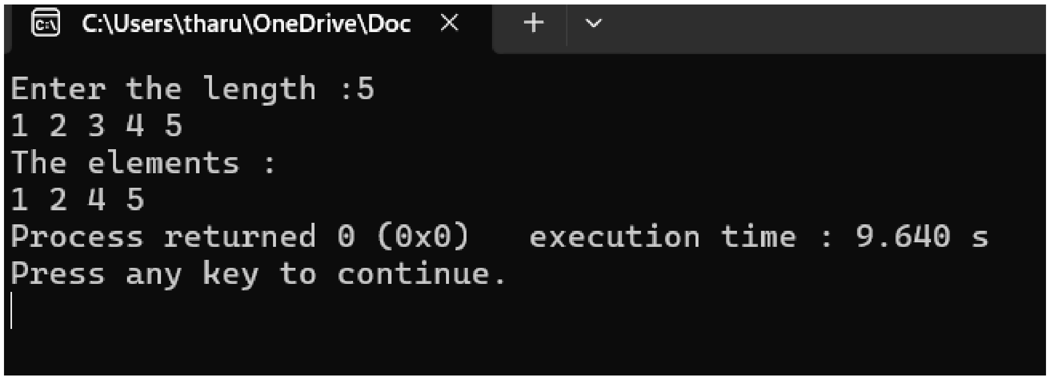
Time Complexity : O(N*M)

Space Complexity : O(M)

17.Delete middle element of a stack Given a stack with push(), pop(), and empty() operations, The task is to delete the middle element of it without using any additional data structure.

```
#include <iostream>
#include <stack>
#include <vector>
using namespace std;
void remove_middle(stack<int>& stack,int idx,int mid){
    if(idx==mid){
        stack.pop();
        return;
    }
    int val=stack.top();
    stack.pop();
    remove_middle(stack,idx+1,mid);
    stack.push(val);
}
int main(){
    int len;
    cout<<"Enter the length :";
    cin>>len;
    if(len==0) cout<<"Operation Cannot be done";
    else{
        int arr[len-1];
        stack<int> stack;
        for(int i=0;i<len;i++){
            int a;
            cin>>a;
            stack.push(a);
        }
        int n=len-1;
        int mid=len/2;
        remove_middle(stack,0,mid);
        while(!stack.empty()){
            arr[n-1]=stack.top();
            stack.pop();
            n--;
        }
        cout<<"The elements :"<<endl;
        for(auto i:arr){
            cout<<i<<" ";
        }
    }
    return 0;
}
```

Output:



```
Enter the length :5
1 2 3 4 5
The elements :
1 2 4 5
Process returned 0 (0x0)   execution time : 9.640 s
Press any key to continue.
```

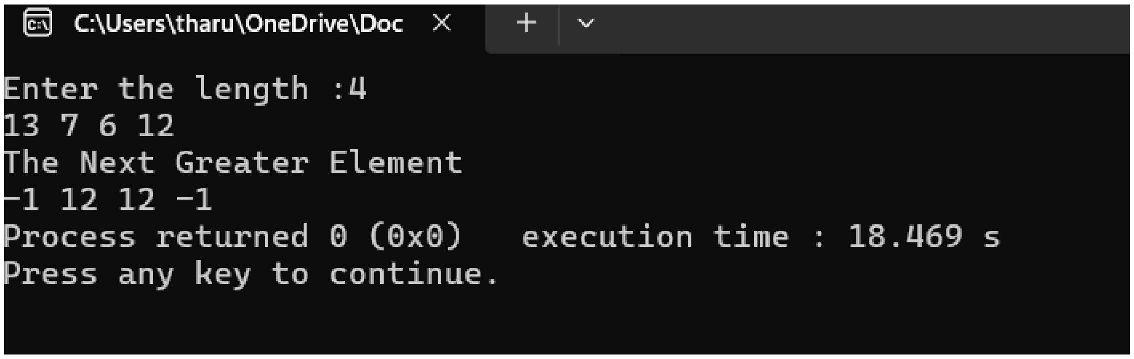
Time Complexity : O(N)

Space Complexity : O(N)

18. Next Greater Element (NGE) for every element in given Array Given an array, print the Next Greater Element (NGE) for every element.

```
#include <iostream>
#include <stack>
using namespace std;
int main(){
   int len;
   cout<<"Enter the length :";
   cin>>len;
   int arr[len];
   for(int i=0;i<len;i++){
      cin>>arr[i];
   }
   stack<pair<int,int>> stack;
   for(int i=0;i<len;i++){
      while(!stack.empty() && stack.top().first<arr[i]){
         auto p=stack.top();
         stack.pop();
         arr[p.second]=arr[i];
      }
      stack.push({arr[i],i});
   }
   while(!stack.empty()){
      arr[stack.top().second]=-1;
      stack.pop();
   }
   cout<<"The Next Greater Element "<<endl;
   for(int i=0;i<len;i++){
      cout<<arr[i]<<" ";
   }
   return 0;
}
```

Output:



```
Enter the length :4
13 7 6 12
The Next Greater Element
-1 12 12 -1
Process returned 0 (0x0)   execution time : 18.469 s
Press any key to continue.
```

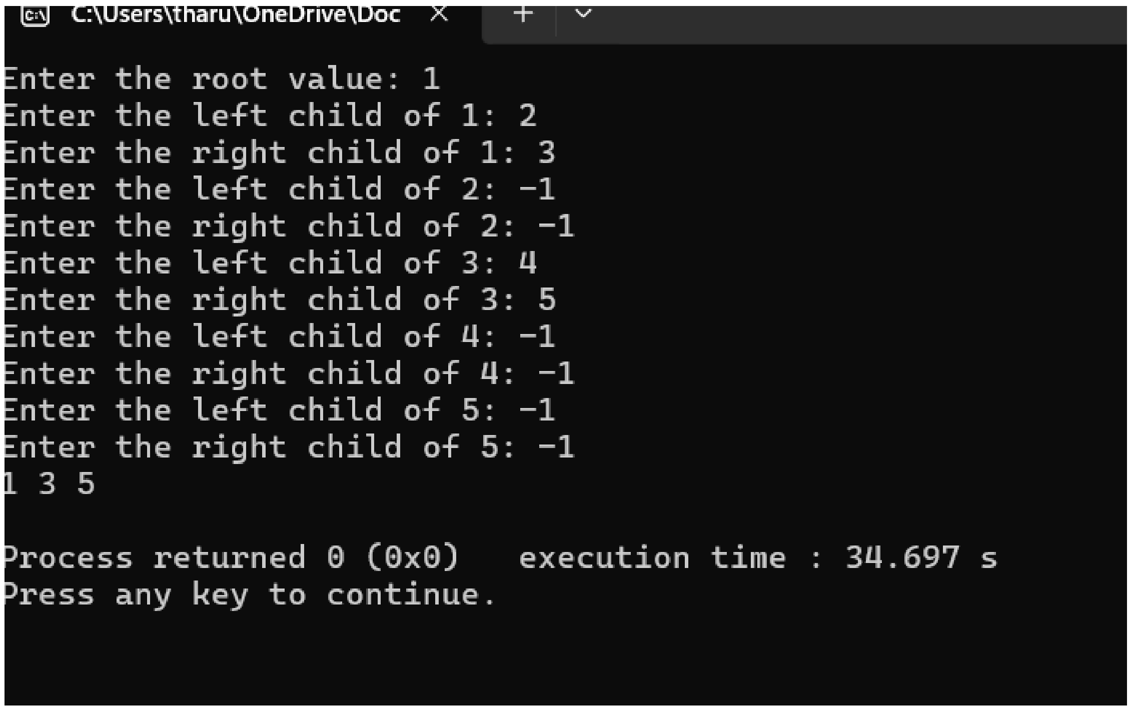
Time Complexity : O(N)

Space Complexity : O(N)

19.Print Right View of a Binary Tree Given a Binary Tree, the task is to print the Right view of it. The right view of a Binary Tree is a set of rightmost nodes for every level.

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

class TreeNode {
public:
    int val;
    TreeNode *left, *right;

    TreeNode(int val) : val(val), left(nullptr), right(nullptr) {}
};

TreeNode* createTree() {
    int rootval;
    cout << "Enter the root value: ";
    cin >> rootval;
    if (rootval == -1) return nullptr;

    TreeNode* root = new TreeNode(rootval);
    queue<TreeNode*> q;
    q.push(root);

    while (!q.empty()) {
        TreeNode* curr = q.front();
        q.pop();

        int leftval, rightval;
        cout << "Enter the left child of " << curr->val << ": ";
        cin >> leftval;
        if (leftval != -1) {
            curr->left = new TreeNode(leftval);
            q.push(curr->left);
        }

        cout << "Enter the right child of " << curr->val << ": ";
        cin >> rightval;
        if (rightval != -1) {
            curr->right = new TreeNode(rightval);
            q.push(curr->right);
        }
    }
    return root;
}

void rightView(TreeNode* curr, vector<int>& result, int currDepth) {
    if (curr == nullptr) return;
    if (currDepth == result.size()) {
        result.push_back(curr->val);
    }

    rightView(curr->right, result, currDepth + 1);
    rightView(curr->left, result, currDepth + 1);
}

vector<int> rightSideView(TreeNode* root) {
    vector<int> result;
    rightView(root, result, 0);
    return result;
}

int main() {
    TreeNode* root = createTree();
    vector<int> result = rightSideView(root);
    for (int val : result) {
        cout << val << " ";
    }
    cout << endl;
    return 0;
}
```

Output:



```
C:\Users\tharu\OneDrive\Doc
Enter the root value: 1
Enter the left child of 1: 2
Enter the right child of 1: 3
Enter the left child of 2: -1
Enter the right child of 2: -1
Enter the left child of 3: 4
Enter the right child of 3: 5
Enter the left child of 4: -1
Enter the right child of 4: -1
Enter the left child of 5: -1
Enter the right child of 5: -1
1 3 5

Process returned 0 (0x0)   execution time : 34.697 s
Press any key to continue.
```

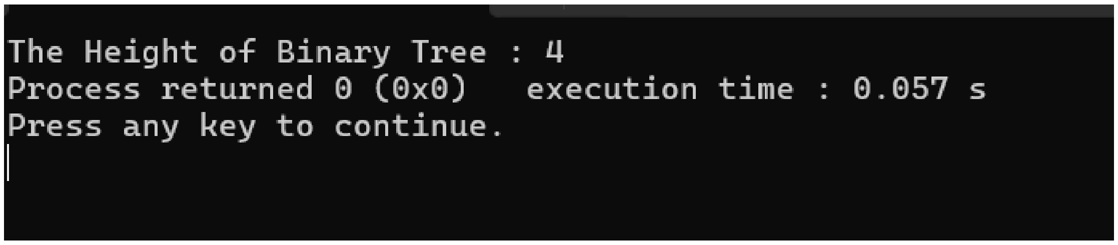
Time compexity : O(N)

Space compexity : O(H)

20.Maximum Depth or Height of Binary Tree Given a binary tree, the task is to find the maximum depth or height of the tree. The height of the tree is the number of vertices in the tree from the root to the deepest node.

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

struct node {
  int data;
  struct node *left;
  struct node *right;
};


struct node *newNode(int data) {
  struct node *node = (struct node *)malloc(sizeof(struct node));

  node->data = data;

  node->left = NULL;
  node->right = NULL;
  return (node);
}

int helper(node* root){
  if(!root) return 0;
  int left=1+helper(root->left);
  int right=1+helper(root->right);
  return max(left,right);
}

int main(){

    struct node* root = newNode(1);
    root->left = newNode(2);
    root->right = newNode(3);
    root->left->left = newNode(4);
    root->right->right = newNode(5);
    root->right->right->left = newNode(6);
    root->right->right->right = newNode(7);
    cout<<"The Height of Binary Tree : "<<helper(root);

    return 0;

}
```

Output:



```
The Height of Binary Tree : 4
Process returned 0 (0x0)   execution time : 0.057 s
Press any key to continue.
```

Time compexity : O(N)

Space compexity : O(H)