Ques 1) Write a function to reverse a singly linked list. The function should take the head of the list and return the new head of the reversed list.

**CODE:**

#include<iostream>

using namespace std;

struct node

{

int data;

struct node\* link;

};

node\* addnode(node\* head)

{

int value;

node\* temp=new node;

cout<<"Enter value: ";

cin>>value;

temp->data=value;

temp->link=nullptr;

if(head==nullptr) head=temp;

else

{

node\* move=head;

while(move->link!=nullptr) move=move->link;

move->link=temp;

}

return head;

}

node\* reverse(node\* head)

{

if (!head || !head->link) return head;

node\* q1=head;

node\* q2=q1->link;

node\* q3=nullptr;

q1->link=nullptr;

while(q2!=nullptr)

{

q3=q2->link;

q2->link=q1;

q1=q2;

q2=q3;

}

head=q1;

return head;

}

void print(node \*head)

{

node \*move=head;

while(move!=nullptr)

{

cout<<move->data<<" ";

move=move->link;

}

cout<<endl;

}

int main()

{

int ch;

node\* head = nullptr;

do

{

cout << "\*\*\*\*\*\*\*\*\*\* MENU \*\*\*\*\*\*\*\*\*\*\n1. Add node\n2. Reverse\nEnter choice: ";

cin>>ch;

switch (ch)

{

case 1:

head = addnode(head);

break;

case 2:

head = reverse(head);

print(head);

break;

default:

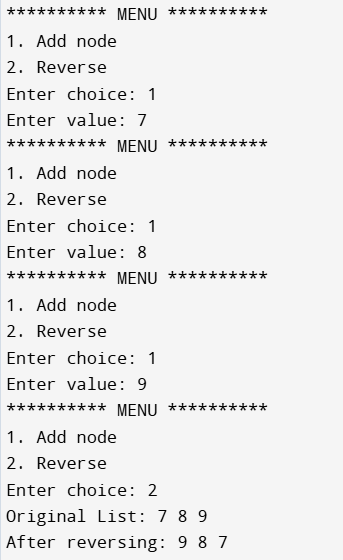
exit(0);

}

} while (true);

return 0;

}



Ques 2) Given a string, find the length of the longest substring without repeating characters. The function should return an integer representing the length of the longest substring without repeating characters.

CODE:

#include<iostream>

#include<vector>

#include<algorithm>

#include<map>

using namespace std;

int longest(string s)

{

map<char,int>mp;

int i=0,j=0,max=0;

while(j<s.length( ))

{

mp[s[j]]++;

if(mp[s[j]]>1)

{

while(i<=j && mp[s[j]]>1)

{

mp[s[i]]--;

i++;

}

}

if(max<j-i+1)

max=j-i+1;

j++;

}

return max;

}

int main( )

{

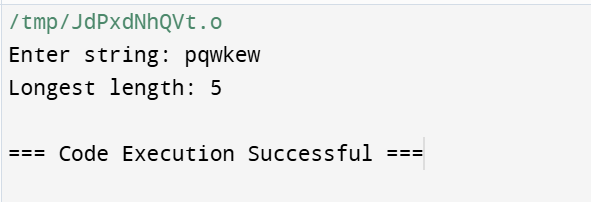
string s;

cout<<"Enter string: ";

cin>>s;

cout<<"Longest length: "<<longest(s);

}



Ques 3) Given a non-empty binary tree, find the maximum path sum. A path is defined as any sequence of nodes from some starting node to any node in the tree along the parent-child connections. The path must contain at least one node and does not need to go through the root. The function should return an integer representing the maximum path sum.

**CODE:**

#include<iostream>

#include<queue>

#include<limits.h>

using namespace std;

struct TreeNode

{

int data;

TreeNode \*left,\*right;

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

};

TreeNode \*insertNode(TreeNode \*root, int data)

{

if(!root) return new TreeNode(data);

queue<TreeNode\*>q;

q.push(root);

while(!q.empty( ))

{

TreeNode \*temp=q.front( );

q.pop( );

if(!temp->left)

{

temp->left=new TreeNode(data);

break;

}

else q.push(temp->left);

if(!temp->right)

{

temp->right=new TreeNode(data);

break;

}

else q.push(temp->right);

}

return root;

}

int dfs(TreeNode\* root, int& maxi)

{

if(root==nullptr) return 0;

int left=max(dfs(root->left,maxi),0);

int right=max(dfs(root->right,maxi),0);

int total=root->data+left+right;

maxi=max(total,maxi);

return root->data+max(left,right);

}

int maxPathSum(TreeNode\* root)

{

int maxi = INT\_MIN;

dfs(root, maxi);

return maxi;

}

int main( )

{

TreeNode \*root=nullptr;

int n,data;

cout<<"Enter num of nodes: ";

cin>>n;

cout<<"Enter values: ";

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

{

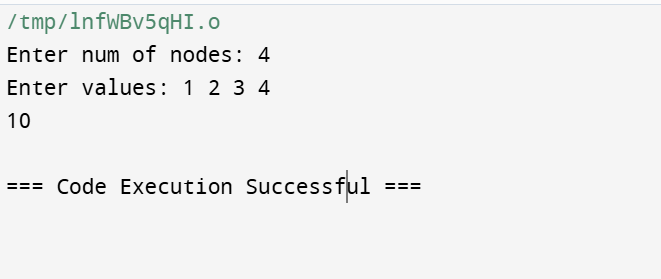
cin>>data;

root=insertNode(root,data);

}

cout<<maxPathSum(root);

}



Ques 4) Design an algorithm to serialize and deserialize a binary tree. Serialization is the process of converting a data structure or object into a sequence of bits so that it can be stored in a file or memory buffer, or transmitted across a network connection link to be reconstructed later in the same or another computer environment. Implement the serialize and deserialize methods.

**CODE:**

Ques 5) Write a function to rotate an array to the right by k steps. The function should modify the array in place to achieve the rotation.

CODE:

#include<iostream>

#include<vector>

#include<algorithm>

#include<map>

using namespace std;

void rotate(vector<int>& nums, int k)

{

vector<int>ans;

int i=nums.size( )-1;

int count=0;

k=k%nums.size( );

while(i>=0 && count<k)

{

ans.push\_back(nums[i]);

nums.pop\_back();

count++;

i--;

}

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

nums.insert(nums.begin( )+0,ans[i]);

}

int main( )

{

int num,k;

cout<<"Enter number of elements: ";

cin>>num;

vector<int>arr(num);

cout<<"Enter elements: ";

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

cin>>arr[i];

cout<<"Enter num of steps: ";

cin>>k;

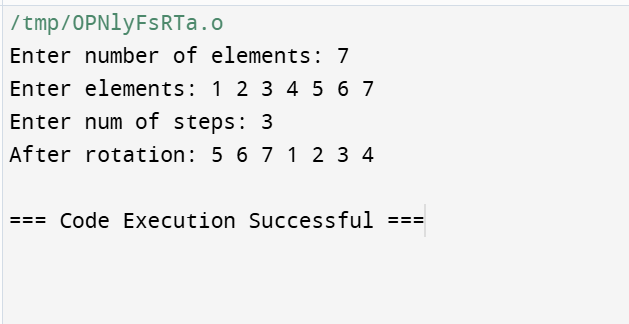
rotate(arr,k);

cout<<"After rotation: ";

for(auto it: arr)

cout<<it<<" ";

}



Ques 6) Write a function to find the factorial of a given number. The function should return the factorial of the number.

**CODE:**

#include<iostream>

using namespace std;

int fact(int num)

{

if(num==0 || num==1)

return num;

return num\*fact(num-1);

}

int main( )

{

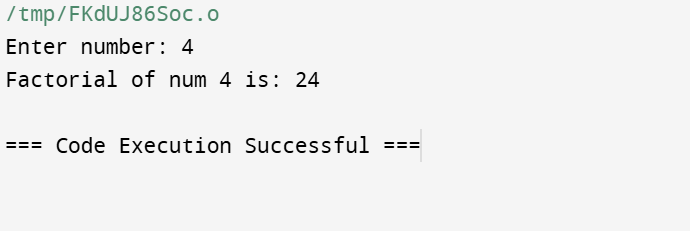
int num;

cout<<"Enter number: ";

cin>>num;

cout<<"Factorial of num "<<num<<" is: "<<fact(num);

}



Ques 7) Write a function to compute the sum of the digits of a given number. The function should return the sum of the digits of the number.

**CODE:**

#include<iostream>

using namespace std;

int sum(int num)

{

int ans=0;

while(num!=0)

{

ans+=num%10;

num/=10;

}

return ans;

}

int main( )

{

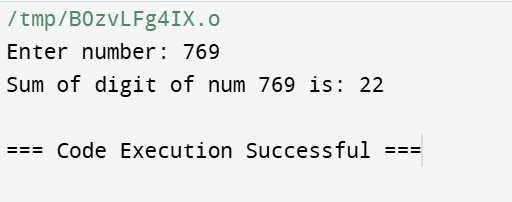
int num;

cout<<"Enter number: ";

cin>>num;

cout<<"Sum of digit of num "<<num<<" is: "<<sum(num);

}



Ques 8) Write a function to find the greatest common divisor (GCD) of two numbers. The function should return the GCD of a and b.

**CODE:**

#include<iostream>

#include<algorithm>

using namespace std;

int lcm(int n1,int n2)

{

int num1=n1,num2=n2;

int ans=1;

int i=2;

while(n1!=1 || n2!=1)

{

if(n1%i==0 || n2%i==0)

{

ans\*=i;

if(n1%i==0)

n1/=i;

if(n2%i==0)

n2/=i;

}

else

i++;

}

return (num1\*num2)/ans;

}

int main( )

{

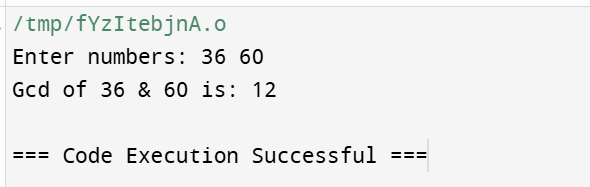
int n1,n2;

cout<<"Enter numbers: ";

cin>>n1>>n2;

cout<<"Gcd of "<<n1<<" & "<<n2<<" is: "<<lcm(n1,n2);

}



Ques 9) Write a function to find the maximum difference between any two elements in an array. The function should return the maximum difference between any two elements in the array.

CODE:

#include<iostream>

#include<vector>

#include<algorithm>

using namespace std;

int main( )

{

int n;

cout<<"Enter number of elements in an array: ";

cin>>n;

vector<int>arr(n);

printf("Enter numbers: ");

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

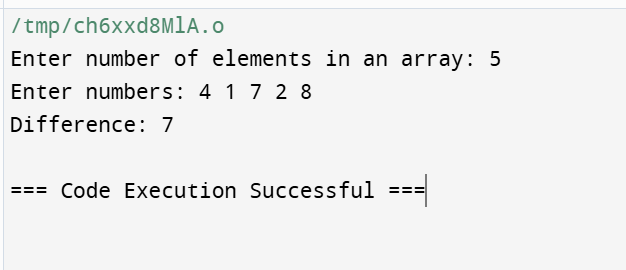
cin>>arr[i];

int mini=\*min\_element(arr.begin( ),arr.end( ));

int maxi=\*max\_element(arr.begin( ),arr.end( ));

cout<<"Difference: "<<maxi-mini;

}



Ques 10) Write a function to check if a given string contains only alphabetic characters. The function should return true if the string contains only alphabetic characters, and false otherwise.

CODE:

#include<iostream>

#include<vector>

#include<algorithm>

using namespace std;

bool check(string s)

{

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

{

if(!isalpha(s[i]))

return false;

}

return true;;

}

int main( )

{

string s;

cout<<"Enter string: ";

cin>>s;

int ans=check(s);

if(ans==1)

cout<<"true";

else

cout<<"false";

}

