**Questions**

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.

#include <iostream>

#include <bits/stdc++.h>

#include <vector>

using namespace std;

struct linked\_list{

    int val;

    linked\_list\* next;

    linked\_list()

    {

        val=0;

        next=NULL;

    }

    linked\_list(int x)

    {

        val=x;

        next=NULL;

    }

    linked\_list\* push(int x)

    {

        linked\_list\* curr=this;

        curr->next=new linked\_list(x);

        curr=curr->next;

        return curr;

    }

};

linked\_list\* reverse(linked\_list\* head)

{

    linked\_list\* nextNode=new linked\_list();

    linked\_list\* prevNode=NULL;

    linked\_list\* curr=head;

    while(curr!=NULL)

    {

        nextNode=curr->next;

        curr->next=prevNode;

        prevNode=curr;

        curr=nextNode;

    }

    head=prevNode;

    return head;

}

int main()

{

    cout<<"Length of the linked list:";

    int N;

    cin>>N;

    cout<<"Input of the linked list:"<<endl;

    int x;

    cin>>x;

    linked\_list\* head=new linked\_list(x);

    linked\_list\* curr=head;

    for(int t=1;t<N;t++)

    {

        cin>>x;

        curr=curr->push(x);

    }

    head=reverse(head);

    curr=head;

    while(curr!=NULL)

    {

        cout<<curr->val<<" ";

        curr=curr->next;

    }

}

A black screen with white text

Description automatically generated

1. 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.

#include <iostream>

#include <bits/stdc++.h>

#include <vector>

#include <string>

#include <map>

using namespace std;

int longest\_substring(string s)

{

    int n=s.size();

    unordered\_map<char,int> mp;

    int left=0,right=0;

    int ans=0;

    while(right<n)

    {

        while(mp[s[right]]==1)

        {

            mp[s[left]]--;

            left++;

        }

        ans=max(ans,right-left+1);

        mp[s[right]]++;

        right++;

    }

    return ans;

}

int main()

{

    string input;

    cout<<"Input String: ";

    cin>>input;

    int result=longest\_substring(input);

    cout<<"The Length of the longest substring without repitition: "<<result;

}

A black background with white text

Description automatically generated

1. 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.

#include <iostream>

#include <bits/stdc++.h>

#include <vector>

#include <string>

#include <map>

#include <queue>

using namespace std;

struct TreeNode {

    int val;

    TreeNode\* left;

    TreeNode\* right;

    TreeNode(){

        val=0;

        left=nullptr;

        right=nullptr;

    }

    TreeNode(int x){

        val=x;

        left=nullptr;

        right=nullptr;

    }

    TreeNode(int x, TreeNode\* l, TreeNode\* r){

        val=x;

        left=l;

        right=r;

    }

};

void levelOrder(TreeNode\* root)

{

    if (root == NULL)

        return;

    queue<TreeNode\*> q;

    q.push(root);

    while (!q.empty())

    {

        if(q.front()==NULL)

        {

            cout<<"NULL ";

            q.pop();

            continue;

        }

        cout << q.front()->val << " ";

        q.push(q.front()->left);

        q.push(q.front()->right);

        q.pop();

    }

}

/// Here is the actual code ---------------------//

int maximum=0;

int recur(TreeNode\* root){

    if(root==NULL)

    {

        cout<<"@";

        return 0;

    }

    int a1=0;

    int a2=0;

    if(root->left!=NULL)

    a1=recur(root->left);

    if(root->right!=NULL)

    a2=recur(root->right);

    int ans=max(a1,a2);

    ans=max(root->val,ans+root->val);

    int a21=max(ans,a1+root->val+a2);

    maximum=max(maximum,a21);

    return ans;

}

int maxPathSum(TreeNode\* root) {

    maximum = root->val;

    int x=recur(root);

    return maximum;

}

//------------------------------------------------------------//

int main()

{

    // The array is in level order traversal

    vector<int> a={-10,9,20,NULL,NULL,9,7}; //root should not be NULL which is a[0]

    //------------------------------//

    int n=a.size();

    TreeNode\* root=new TreeNode(a[0]);

    queue<TreeNode\*> q;

    q.push(root);

    for(int i=1;i<n;i+=2)

    {

        if(a[i]!=NULL)//left

        {

            TreeNode\* temp=new TreeNode(a[i]);

            q.front()->left=temp;

            q.push(temp);

        }

        if(i+1<n && a[i+1]!=NULL)

        {

            TreeNode\* temp=new TreeNode(a[i+1]);

            q.front()->right=temp;

            q.push(temp);

        }

        q.pop();

    }

    cout<<"Tree in level order Traversal"<<endl;

    levelOrder(root);

    cout<<endl;

    cout<<"Max path sum: "<<maxPathSum(root);

}

A black background with white text

Description automatically generated

1. 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.

#include <iostream>

#include <bits/stdc++.h>

#include <vector>

#include <string>

#include <map>

#include <queue>

using namespace std;

struct TreeNode {

    int val;

    TreeNode\* left;

    TreeNode\* right;

    TreeNode(){

        val=0;

        left=nullptr;

        right=nullptr;

    }

    TreeNode(int x){

        val=x;

        left=nullptr;

        right=nullptr;

    }

    TreeNode(int x, TreeNode\* l, TreeNode\* r){

        val=x;

        left=l;

        right=r;

    }

};

void levelOrder(TreeNode\* root)

{

    if (root == NULL)

        return;

    queue<TreeNode\*> q;

    q.push(root);

    while (!q.empty())

    {

        if(q.front()==NULL)

        {

            cout<<"NULL ";

            q.pop();

            continue;

        }

        cout << q.front()->val << " ";

        q.push(q.front()->left);

        q.push(q.front()->right);

        q.pop();

    }

    cout<<endl;

}

/// Here is the actual code ---------------------//

void serialize(TreeNode\* root,FILE\* fp)

{

    if(root==NULL)

    {

        fprintf(fp,"%d ",-1);

        return;

    }

    fprintf(fp,"%d ",root->val);

    serialize(root->left,fp);

    serialize(root->right,fp);

}

void deserialize(TreeNode\*& root,FILE\* fp)

{

    int val;

    if(!(fscanf(fp,"%d ",&val)) || val==-1)

    {

        return;

    }

    root=new TreeNode(val);

    deserialize(root->left,fp);

    deserialize(root->right,fp);

}

//------------------------------------------------------------//

int main()

{

    // The array is in level order traversal and only positive integer, as we use -1 as marker for NULL in tree

    vector<int> a={10,9,20,NULL,NULL,9,7}; //root should not be NULL which is a[0]

    //------------------------------//

    int n=a.size();

    TreeNode\* root=new TreeNode(a[0]);

    queue<TreeNode\*> q;

    q.push(root);

    for(int i=1;i<n;i+=2)

    {

        if(a[i]!=NULL)//left

        {

            TreeNode\* temp=new TreeNode(a[i]);

            q.front()->left=temp;

            q.push(temp);

        }

        if(i+1<n && a[i+1]!=NULL)

        {

            TreeNode\* temp=new TreeNode(a[i+1]);

            q.front()->right=temp;

            q.push(temp);

        }

        q.pop();

    }

    cout<<"Given Tree in level order Traversal"<<endl;

    levelOrder(root);

    //file open to serialize the tree

    // 4\_data.txt should be there in the current directory

    FILE\* fp = fopen("4\_data.txt", "w");

    if (fp == NULL) {

        puts("Could not open file");

        return 0;

    }

    serialize(root, fp);

    fclose(fp);

    //deserialize tree from the file

    TreeNode\* root1 = NULL;

    fp = fopen("4\_data.txt", "r");

    deserialize(root1, fp);

    cout<<"Level Traversal of the tree constructed from file:\n";

    levelOrder(root1);

    fclose(fp);

}

Data file output: A black background with white text

Description automatically generated

Terminal Ouput:A black screen with white text

Description automatically generated

1. 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.

#include <iostream>

#include <bits/stdc++.h>

#include <vector>

#include <algorithm>

using namespace std;

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

{

    int n=vec.size();

    k=k%n;

    reverse(vec.begin(),vec.begin()+k);

    reverse(vec.begin()+k+1,vec.end());

    reverse(vec.begin(),vec.end());

}

int main()

{

    cout<<"Length of the array:";

    int N;

    cin>>N;

    cout<<"Input of the array:"<<endl;

    vector<int> vec(N);

    for(int t=0;t<N;t++)

    {

        cin>>vec[t];

    }

    cout<<"Number steps to rotate by: ";

    int k;

    cin>>k;

    rotate(vec,k);

    cout<<"Rotated array: ";

    for(int x:vec)

    {

        cout<<x<<" ";

    }

}

A black screen with white text

Description automatically generated

1. Write a function to find the factorial of a given number.The function should return the factorial of the number.

#include <iostream>

#include <bits/stdc++.h>

using namespace std;

long long int fact(int x)

{

    if(x==0 || x==1)

    {

        return 0;

    }

    long long int out=1;

    for(int i=2;i<=x;i++)

    {

        out\*=i;

    }

    return out;

}

int main()

{

    // can find factorial upto x=19

    cout<<"Number whose factorial you want to find: ";

    int x;

    cin>>x;

    if(x<0)

    {

        cout<<"Error: Negative Number input";

        return 0;

    }

    long long int f=fact(x);

    cout<<"The factorial is: "<<f;

}



1. 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.

#include <iostream>

#include <bits/stdc++.h>

using namespace std;

int digit\_sum(string x)

{

    int sum=0;

    for(int i=0;i<x.size();i++)

    {

        int d=(int)(x[i]-'0');

        sum+=d;

    }

    return sum;

}

int main()

{

    cout<<"Number whose sum of digits you want to find: ";

    string x;

    cin>>x;

    int f=digit\_sum(x);

    cout<<"The sum of digits is: "<<f;

}



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

#include <iostream>

#include <bits/stdc++.h>

using namespace std;

int gcd(int a,int b)

{

    while(a>0 && b>0)

    {

        if(a<b)

        {

            swap(a,b);

        }

        if(a%b==0)

        {

            return b;

        }

        int x=a-b;

        a=b;

        b=x;

    }

    if(b==0)

    {

        return a;

    }

    return b;

}

int main()

{

    cout<<"The two numbers whose GCD you want: ";

    int a,b;

    cin>>a;

    cin>>b;

    cout<<"GCD of "<<a<<" and "<<b<<" is: "<<gcd(a,b);

}

A black background with white text

Description automatically generated

1. 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.

#include <iostream>

#include <bits/stdc++.h>

#include <vector>

using namespace std;

int max\_dif(vector<int> vec)

{

    int m1=vec[0];

    int m2=vec[0];

    for(int x:vec)

    {

        m1=min(m1,x);

        m2=max(m2,x);

    }

    return (m2-m1);

}

int main()

{

    cout<<"Length of the array: ";

    int N;

    cin>>N;

    cout<<"Input Array: "<<endl;

    vector<int> vec(N);

    for(int t=0;t<N;t++)

    {

        cin>>vec[t];

    }

    cout<<"Maximum difference any 2 elements is: "<<max\_dif(vec);

}

A black background with white text

Description automatically generated

1. 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.

#include <iostream>

#include <bits/stdc++.h>

#include <vector>

using namespace std;

bool check(char& c)

{

    if((c>='a'&&c<='z')||(c>='A' && c<='Z'))

        return true;

    return false;

}

int main()

{

    cout<<"Input String: ";

    string s;

    cin>>s;

    for(char c:s)

    {

        if(!check(c))

        {

            cout<<"Not all alphabet string";

            return 0;

        }

    }

    cout<<"All alphabet string";

}

