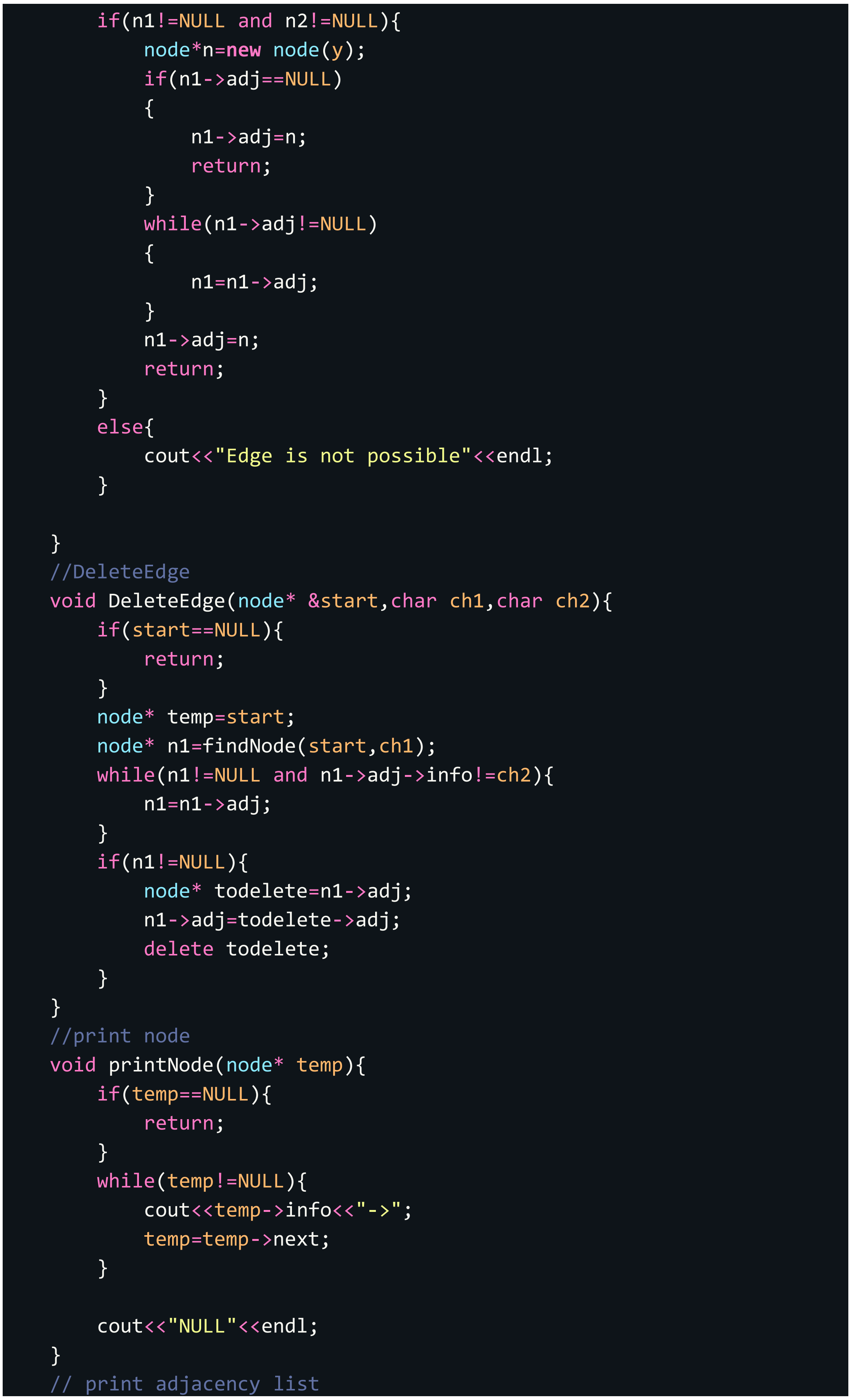
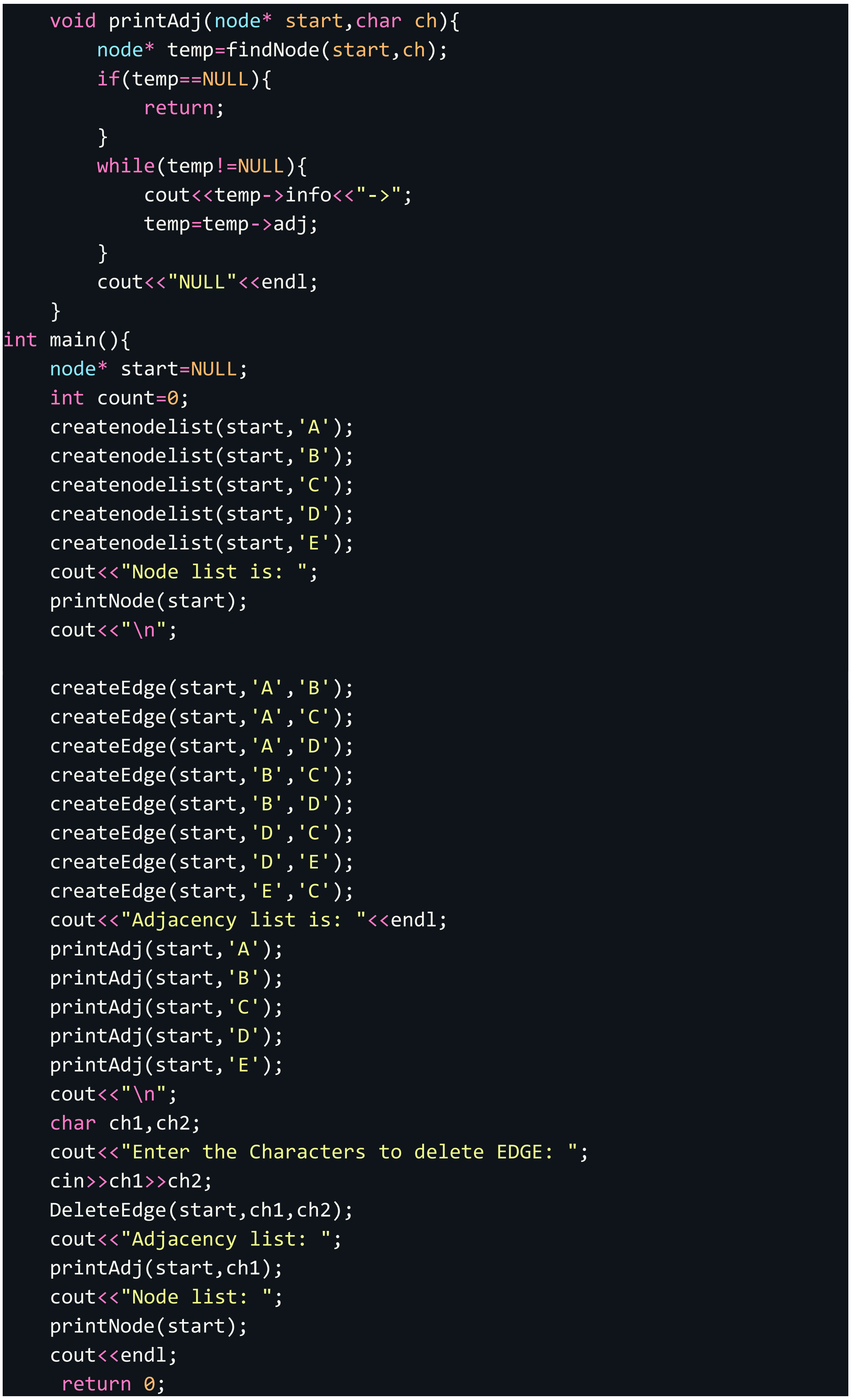
SourceCode

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

class node{
public:
    char info;
    node* next;
    node* adj;
    node(char val){
        info=val;
        next=NULL;
        adj=NULL;
    }
};
 // Find loc
    node* findNode(node* start,char ch){
        node* temp=start;
        if(start==NULL){
            return NULL;
        }
        while(temp!=NULL){
            if(temp->info==ch){
                return temp;
            }
            temp=temp->next;
        }
        return NULL;
    }
    //createnode insertAtLast
    void createnodelist(node* &start,char ele){
        node* n=new node(ele);
        if(start==NULL){
            start=n;
            return;
        }
                node*temp=start;

        while(temp->next!=NULL){
            temp=temp->next;
        }
        temp->next=n;
        return;
    }
    // create edge
    void createEdge(node* &start,char x,char y){
        node* temp=start;
        node* n1=findNode(temp,x);
        node* n2=findNode(temp,y);
```

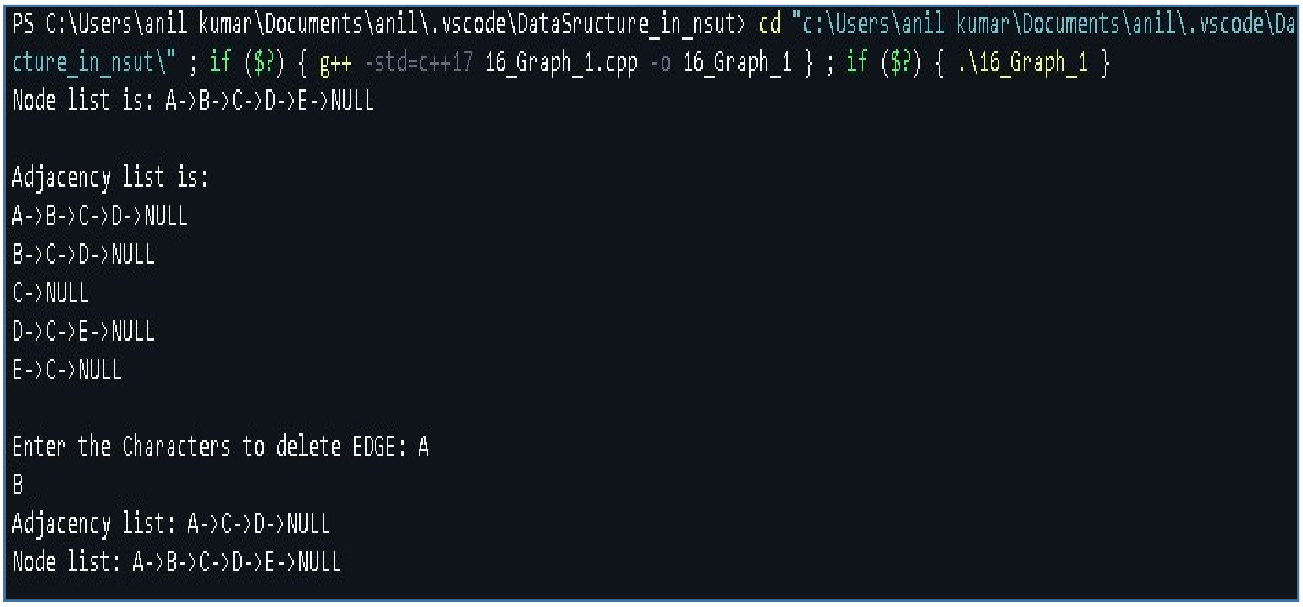


```
    if(n1!=NULL and n2!=NULL){
        node*n=new node(y);
        if(n1->adj==NULL)
        {
            n1->adj=n;
            return;
        }
        while(n1->adj!=NULL)
        {
            n1=n1->adj;
        }
        n1->adj=n;
        return;
    }
    else{
        cout<<"Edge is not possible"<<endl;
    }

}
//DeleteEdge
void DeleteEdge(node* &start,char ch1,char ch2){
    if(start==NULL){
        return;
    }
    node* temp=start;
    node* n1=findNode(start,ch1);
    while(n1!=NULL and n1->adj->info!=ch2){
        n1=n1->adj;
    }
    if(n1!=NULL){
        node* todelete=n1->adj;
        n1->adj=todelete->adj;
        delete todelete;
    }
}
//print node
void printNode(node* temp){
    if(temp==NULL){
        return;
    }
    while(temp!=NULL){
        cout<<temp->info<<"->";
        temp=temp->next;
    }

    cout<<"NULL"<<endl;
}
// print adjacency list
```



```cpp
    void printAdj(node* start,char ch){
        node* temp=findNode(start,ch);
        if(temp==NULL){
            return;
        }
        while(temp!=NULL){
            cout<<temp->info<<"->";
            temp=temp->adj;
        }
        cout<<"NULL"<<endl;
    }
int main(){
    node* start=NULL;
    int count=0;
    createnodelist(start,'A');
    createnodelist(start,'B');
    createnodelist(start,'C');
    createnodelist(start,'D');
    createnodelist(start,'E');
    cout<<"Node list is: ";
    printNode(start);
    cout<<"\n";

    createEdge(start,'A','B');
    createEdge(start,'A','C');
    createEdge(start,'A','D');
    createEdge(start,'B','C');
    createEdge(start,'B','D');
    createEdge(start,'D','C');
    createEdge(start,'D','E');
    createEdge(start,'E','C');
    cout<<"Adjacency list is: "<<endl;
    printAdj(start,'A');
    printAdj(start,'B');
    printAdj(start,'C');
    printAdj(start,'D');
    printAdj(start,'E');
    cout<<"\n";
    char ch1,ch2;
    cout<<"Enter the Characters to delete EDGE: ";
    cin>>ch1>>ch2;
    DeleteEdge(start,ch1,ch2);
    cout<<"Adjacency list: ";
    printAdj(start,ch1);
    cout<<"Node list: ";
    printNode(start);
    cout<<endl;
     return 0;
}
```

Output



```
PS C:\Users\anil kumar\Documents\anil\.vscode\DataSructure_in_nsut> cd "c:\Users\anil kumar\Documents\anil\.vscode\Da
cture_in_nsut\" ; if ($?) { g++ -std=c++17 16_Graph_1.cpp -o 16_Graph_1 } ; if ($?) { .\16_Graph_1 }
Node list is: A->B->C->D->E->NULL

Adjacency list is:
A->B->C->D->NULL
B->C->D->NULL
C->NULL
D->C->E->NULL
E->C->NULL

Enter the Characters to delete EDGE: A
B
Adjacency list: A->C->D->NULL
Node list: A->B->C->D->E->NULL
```