#include<iostream>

Source Code

using namespace std;

class node{

public:

    int data;

    node\* prev,\*next;

    node(int val){

        data=val;

        prev=NULL;

        next=NULL;

    }

};

void insertAtHead(node\* &head,int val){

    node\* n=**new** node(val);

    if(head==NULL){

        head=n;

        return;

    }

    if(val<head->data){

        head->prev=n;

        n->next=head;

        head=n;

    }

    else{

        cout<<"Value:"<<val<<" is greater than head"<<endl;

    }

}

void insertAtTail(node\* head,int val){

    if(head==NULL){

        insertAtHead(head,val);

    }

    node\* n=**new** node(val);

    node\* temp=head;

    while(temp->next!=NULL){

        temp=temp->next;

    }

    if(val>temp->data){

        temp->next=n;

        n->prev=temp;

    }

    else{

        cout<<"Value:"<<val<<" is Smaller than Tail"<<endl;

    }

}

void insertEle(node\* &head,int val){  //INSERT VALUE

    if(head==NULL){

        insertAtHead(head,val);

        return;

    }

    node\* n=**new** node(val);

    node\* temp=head;

    while(temp!=NULL ){

        if(temp->data>val and temp->prev==NULL){

            insertAtHead(head,val);

        }

        if(temp->data<val){

            if(temp->next==NULL){

            insertAtTail(head,val);

            return;

            }

            if(temp->next->data>val){

                node\* newnode=temp->next;

                    temp->next=n;

                    n->prev=temp;

                    newnode->prev=n;

                    n->next=newnode;

            }

        }

        temp=temp->next;

    }

}

void display(node\* head){

    cout<<"Display in Sorted form: ";

    if(head==NULL){

        return;

    }

    while(head!=NULL){

        cout<<head->data<<"->";

        head=head->next;

    }

    cout<<"NULL"<<endl;

}

void display2(node\* head){

    cout<<"Display in reverse: ";

    if(head==NULL){

        return;

    }

    while(head->next!=NULL){

        head=head->next;

    }

    while(head!=NULL){

        cout<<head->data<<"->";

        head=head->prev;

    }

    cout<<"NULL"<<endl;

}

int main(){

node\* head=NULL;

insertEle(head,5);

insertEle(head,3);

insertEle(head,2);

insertEle(head,4);

insertEle(head,9);

insertEle(head,8);

insertEle(head,7);

insertEle(head,1);

display(head);

display2(head);

return 0;

}

OUTPUT

#include<iostream>

SOURCE CODE

using namespace std;

class node{

public:

    int data;

    node\* prev,\*next;

    node(int val){

        data=val;

        prev=NULL;

        next=NULL;

    }

};

void insertAtHead(node\* &head,int val){

    node\* n=**new** node(val);

    if(head==NULL){

        head=n;

        return;

    }

    if(val<head->data){

        head->prev=n;

        n->next=head;

        head=n;

    }

    else{

        cout<<"Value:"<<val<<" is greater than head"<<endl;

    }

}

void insertAtTail(node\* head,int val){

    if(head==NULL){

        insertAtHead(head,val);

    }

    node\* n=**new** node(val);

    node\* temp=head;

    while(temp->next!=NULL){

        temp=temp->next;

    }

    if(val>temp->data){

        temp->next=n;

        n->prev=temp;

    }

    else{

        cout<<"Value:"<<val<<" is Smaller than Tail"<<endl;

    }

}

void insertEle(node\* &head,int val){  //INSERT VALUE

    if(head==NULL){

        insertAtHead(head,val);

        return;

    }

    node\* n=**new** node(val);

    node\* temp=head;

    while(temp!=NULL ){

        if(temp->data>val and temp->prev==NULL){

            insertAtHead(head,val);

        }

        if(temp->data<val){

            if(temp->next==NULL){

            insertAtTail(head,val);

            return;

            }

            if(temp->next->data>val){

                node\* newnode=temp->next;

                    temp->next=n;

                    n->prev=temp;

                    newnode->prev=n;

                    n->next=newnode;

            }

        }

        temp=temp->next;

    }

}

void deleteAtstart(node\* &head){

    if(head==NULL){

        cout<<"List is empty"<<endl;

        return;

    }

    node\* temp=head;

    temp->next->prev=NULL;

    head=temp->next;

    delete temp;

}

void deletion(node\* &head,int val){  //delete At pos

    if(head==NULL){

        cout<<"List is empty"<<endl;

        return;

    }

    if(head->data==val){

        deleteAtstart(head);

    }

    else{

        node\* temp=head;

        while(temp!=NULL and temp->data<val){

            temp=temp->next;

        }

        if(temp==NULL||temp->data>val){

            return;

        }

        temp->prev->next=temp->next;

        if(temp->next!=NULL){

            temp->next->prev=temp->prev;

        }

        delete temp;

    }

}

void display(node\* head){

    cout<<"Display in Sorted form: ";

    if(head==NULL){

        return;

    }

    while(head!=NULL){

        cout<<head->data<<"->";

        head=head->next;

    }

    cout<<"NULL"<<endl;

}

void display2(node\* head){

    cout<<"Display in reverse: ";

    if(head==NULL){

        return;

    }

    while(head->next!=NULL){

        head=head->next;

    }

    while(head!=NULL){

        cout<<head->data<<"->";

        head=head->prev;

    }

    cout<<"NULL"<<endl;

}

int main(){

node\* head=NULL;

insertEle(head,5);

insertEle(head,3);

insertEle(head,2);

insertEle(head,4);

insertEle(head,9);

insertEle(head,8);

insertEle(head,7);

insertEle(head,1);

display(head);

display2(head);

insertAtTail(head,9);

insertAtTail(head,8);

display(head);

cout<<" deletion begin"<<endl;

deletion(head,6);

deletion(head,8);

deletion(head,1);

deletion(head,7);

deletion(head,3);

display(head);

return 0;

OUTPUT

#include<iostream>

SOURCE CODE

using namespace std;

void input(int \*\*arr,int m,int n){

    cout<<"Enter the elements of a matrix: "<<endl;

    for(int i=0;i<m;i++){

        for(int j=0;j<n;j++){

            cin>>arr[i][j];

        }

    }

}

void printM(int \*\*arr,int m,int n){

    cout<<"Matrix is: "<<endl;

    for(int i=0;i<m;i++){

        for(int j=0;j<n;j++){

            cout<<arr[i][j]<<" ";

        }

        cout<<endl;

    }

}

void multiplication(int \*\*a,int \*\*b,int r1,int c1,int r2,int c2){

    int\*\*c;

    if(r2!=c1){

        cout<<"Multiplication is not possible Because row2 is not equal to column 1\n"<<endl;

        return;

    }

    c=**new** int\*[r1];

    for(int i=0;i<r1;i++){

        c[i]=**new** int[c2];

    }

    for(int i=0;i<r1;i++){

        for(int j=0;j<c2;j++){

            c[i][j]=0;

            for(int k=0;k<c1;k++){

                c[i][j]+=a[i][k]\*b[k][j];

            }

        }

    }

    printM(c,r1,c2);

}

int main(){

    int \*\*a,\*\*b,\*\*c;

    int row1,row2,col1,col2;

    cout<<"\*\*\*\*\*\*\*\*Matrix Multiplication\*\*\*\*\*\*\*\*\*"<<endl;

    cout<<"Enter the row and column of 1 Matrix respectively"<<endl;

    cin>>row1>>col1;

    a=**new** int\*[row1];

    for(int i=0;i<row1;i++){

        a[i]=**new** int[col1];

    }

    input(a,row1,col1);

    cout<<"Enter the row and column of 2 Matrix respectively"<<endl;

    cin>>row2>>col2;

    b=**new** int\*[row2];

    for(int i=0;i<row2;i++){

        b[i]=**new** int[col2];

    }

    input(b,row2,col2);

    // //Function for Multiplication

    multiplication(a,b,row1,col1,row2,col2);

return 0;

}

OUTPUT

#include<iostream>

SOURCE CODE

using namespace std;

class stack{

    int \*a;

    int top;

    int cap;

public:

    stack(int val){

        cap=val;

        top=-1;

        a=**new** int[val];

    }

    int isEmpty(){

        if(top==-1){

            return 1;

        }

        return 0;

    }

    int isFull(){

        if(top==cap-1){

            return 1;

        }

        return 0;

    }

    void push(int item){

        if(isFull()){

            cout<<"Stack overflow"<<endl;

            return;

        }

        a[++top]=item;

    }

    int pop(){

        if(isEmpty()){

            cout<<"Stack underflow"<<endl;

            return -1;

        }

        return a[top--];

    }

};

int main(){

    stack ob(4);

    ob.push(1);

    ob.push(2);

    ob.push(3);

    ob.push(4);

    ob.push(5);

    cout<<ob.pop()<<endl;

    cout<<ob.pop()<<endl;

    cout<<ob.pop()<<endl;

    cout<<ob.pop()<<endl;

cout<<ob.pop()<<endl;

    return 0;

}

OUTPUT