**PRAC1 – Double Linked List**

**CODE:**

#include<bits/stdc++.h>

using namespace std;

struct node

{

struct node\* lptr;

int data;

struct node\* rptr;

};

struct node \*Left = NULL;

struct node \*Right = NULL;

void insertleft(int x)

{

struct node \*nn = (struct node \*)malloc(sizeof(struct node));

if(Left == NULL)

{

nn->data = x;

nn->lptr=NULL;

nn->rptr=NULL;

Left = nn;

Right = nn;

}

else

{

nn->data = x;

nn->rptr = Left;

nn->lptr = NULL;

Left->lptr = nn;

Left = nn;

}

}

void insertright(int x)

{

struct node \*nn = (struct node \*)malloc(sizeof(struct node));

if(right == NULL)

{

nn->data = x;

nn->lptr=NULL;

nn->rptr=NULL;

Left = nn;

Right = nn;

}

else

{

nn->data = x;

nn->lptr = Right;

nn->rptr = NULL;

Right->rptr = nn;

Right = nn;

}

}

void deleteleft()

{

if(Left == NULL && Right ==NULL)

{

cout<<"LinkedList UnderFlow\n";

}

else if(Left==Right)

{

struct node \*temp = (struct node \*)malloc(sizeof(struct node));

Left = NULL;

Right = NULL;

free(temp);

cout<<"\nElement Deleted From Left\n";

}

else

{

struct node \*temp = (struct node \*)malloc(sizeof(struct node));

temp = Left;

Left = Left->rptr;

Left->lptr= NULL;

free(temp);

cout<<"\nElement Deleted From Left\n";

}

}

void deleteright()

{

if(Left == NULL && Right ==NULL)

{

cout<<"LinkedList UnderFlow\n";

}

else if(Left==Right)

{

struct node \*temp = (struct node \*)malloc(sizeof(struct node));

Left = NULL;

Right = NULL;

free(temp);

cout<<"\nElement Deleted From Right\n";

}

else

{

struct node \*temp = (struct node \*)malloc(sizeof(struct node));

temp = Right;

Right = Right->lptr;

Right->rptr= NULL;

free(temp);

cout<<"\nElement Deleted From Right\n";

}

}

void display()

{

cout<<"Linked List\n";

struct node \*temp = (struct node \*)malloc(sizeof(struct node));

temp = Left;

while(temp->rptr != NULL)

{

cout<<temp->data<<" ";

temp = temp->rptr;

}

cout<<temp->data<<" \n";

}

int main()

{

int n;

cout<<"By 22CE024 - Naitik Desai\n";

cout<<"Enter 1 for InsertLeft\n";

cout<<"Enter 2 for InsertRight\n";

cout<<"Enter 3 for DeleteLeft\n";

cout<<"Enter 4 for DeleteRight\n";

cout<<"Enter 5 for Display\n";

cout<<"Enter 6 for Exit\n";

do

{

cout<<"Enter the number: ";

cin>>n;

switch(n)

{

case 1:

int x;

cout<<"Enter the insert element: ";

cin>>x;

insertleft(x);

break;

case 2:

int x1;

cout<<"Enter the insert element: ";

cin>>x1;

insertright(x1);

break;

case 3:

deleteleft();

break;

case 4:

deleteright();

break;

case 5:

display();

break;

case 6:

cout<<"Exit From Doubly LinkedList\n";

break;

}

}while(n>0 && n<6);

}

**OUTPUT:**

**A screenshot of a computer program

Description automatically generatedA screenshot of a computer

Description automatically generated**

**PRAC2 – Insertion Sort List**

**CODE:**

class Solution {

public:

ListNode\* insertionSortList(ListNode\* head) {

if(head==NULL || head->next==NULL)

{

return head;

}

ListNode \*temp=head->next;

//head=head->next;

ListNode \*new1;

head->next=NULL;

while(temp)

{

new1 = temp->next;

ListNode \*prev=NULL;

ListNode \*key=head;

if(temp->val < key->val)

{

temp->next=key;

head=temp;

}

else

{

prev=key;

key=key->next;

while(key)

{

if(temp->val < key->val)

{

prev->next=temp;

temp->next=key;

break;

}

prev=key;

key=key->next;

}

if(key==NULL)

{

prev->next=temp;

temp->next=NULL;

}

}

temp=new1;

}

return head;

}

};

**OUTPUT:**

****