Expt no: 6

**PROGRAM: Implementation of Doubly Linked List**

**Input:**

#include<stdio.h>

#include<stdlib.h>

struct Node

{

int data;

struct Node \*prev;

struct Node \*next;

};

struct Node \*start = NULL;

void InsertAtBeg()

{

int x;

printf("Enter the element to be inserted: ");

scanf("%d", &x);

struct Node \*tmp;

tmp=(struct Node\*)malloc(sizeof(struct Node));

if(start==NULL)

{

tmp->data=x;

tmp->prev=NULL;

tmp->next=NULL;

start=tmp;

printf("%d inserted successfully\n", x);

}

else

{

tmp->data=x;

tmp->prev=NULL;

tmp->next=start;

start->prev=tmp;

start=tmp;

printf("%d inserted successfully\n", x);

}

}

void InsertAtEnd()

{

int x;

printf("Enter the element to be inserted: ");

scanf("%d", &x);

struct Node \*tmp, \*p;

tmp=(struct Node\*)malloc(sizeof(struct Node));

if(start==NULL)

{

tmp->data=x;

tmp->prev=NULL;

tmp->next=NULL;

start=tmp;

printf("%d inserted successfully\n", x);

}

else

{

p=start;

while(p->next!=NULL)

{

p=p->next;

}

tmp->data=x;

tmp->prev=p;

tmp->next=NULL;

p->next=tmp;

printf("%d inserted successfully\n", x);

}

}

void InsertBeforeNode()

{

if(start==NULL)

{

printf("Linked list is empty\n");

return;

}

struct Node \*tmp, \*p;

int x, item;

printf("Enter the element to be inserted: ");

scanf("%d", &x);

printf("Enter the Node before which element is to be inserted: ");

scanf("%d", &item);

if(start->data==item)

{

tmp=(struct Node\*)malloc(sizeof(struct Node));

tmp->data=x;

tmp->prev=NULL;

tmp->next=start;

start->prev=tmp;

start=tmp;

printf("%d inserted successfully before %d\n", x, item);

return;

}

p=start;

while(p->next!=NULL)

{

if(p->next->data==item)

{

tmp=(struct Node\*)malloc(sizeof(struct Node));

tmp->data=x;

tmp->prev=p;

tmp->next=p->next;

p->next=tmp;

p->next->prev=tmp;

printf("%d inserted successfully before %d\n", x, item);

return;

}

p=p->next;

}

printf("%d is not present in the list", item);

return;

}

void InsertAfterNode()

{

if(start==NULL)

{

printf("Linked list is empty\n");

return;

}

struct Node \*tmp, \*p;

int x, item;

printf("Enter the element to be inserted: ");

scanf("%d", &x);

printf("Enter the Node after which element is to be inserted: ");

scanf("%d", &item);

p=start;

p=start;

while(p!=NULL)

{

if(p->data==item)

{

tmp=(struct Node\*)malloc(sizeof(struct Node));

tmp->data=x;

tmp->prev=p;

tmp->next=p->next;

p->next=tmp;

if(p->next!=NULL)

{

p->next->prev=tmp;

}

printf("%d inserted successfully after %d\n", x, item);

return;

}

p=p->next;

}

printf("%d is not present in the list", item);

return;

}

void InsertAtPos()

{

if(start==NULL)

{

printf("Linked list is empty\n");

return;

}

struct Node \*tmp, \*p;

int x, i, pos;

printf("Enter the element to be inserted: ");

scanf("%d", &x);

printf("Enter the position at which element is to be inserted: ");

scanf("%d", &pos);

tmp=(struct Node\*)malloc(sizeof(struct Node));

tmp->data=x;

if(pos==1)

{

tmp->prev=NULL;

tmp->next=start;

start->prev=tmp;

start=tmp;

printf("%d inserted successfully at position %d\n", x, pos);

return;

}

p=start;

for(i=1; i<pos-1 && p!=NULL; i++)

{

p=p->next;

}

if(p==NULL)

{

printf("There are less than %d elements\n", pos);

}

else

{

tmp->prev=p;

tmp->next=p->next;

p->next=tmp;

if(p->next!=NULL)

{

p->next->prev=tmp;

}

printf("%d inserted successfully at position %d\n", x, pos);

}

return;

}

void DeletionAtBeg()

{

struct Node \*p;

if(start==NULL)

{

printf("Linked list is empty\n");

return;

}

else if(start->next==NULL)

{

p=start;

printf("%d has been deleted\n", p->data);

start=NULL;

free(p);

}

else

{

p=start;

printf("%d has been deleted\n", p->data);

p->next->prev=NULL;

start=p->next;

free(p);

}

}

void DeletionAtEnd()

{

struct Node \*p;

if(start==NULL)

{

printf("Linked list is empty\n");

return;

}

else if(start->next==NULL)

{

p=start;

printf("%d has been deleted\n", p->data);

start=NULL;

free(p);

}

else

{

p=start;

while(p->next->next!=NULL)

{

p=p->next;

}

printf("%d has been deleted\n", p->next->data);

free(p->next);

p->next=NULL;

}

}

void DeletionInBet()

{

if(start==NULL)

{

printf("Linked list is empty\n");

return;

}

struct Node \*p, \*tmp;

int x;

printf("Enter the element to be deleted: ");

scanf("%d", &x);

if(start->data==x)

{

tmp=start;

start=start->next;

free(tmp);

printf("Element %d has been deleted\n", x);

return;

}

p=start;

while(p->next!=NULL)

{

if(p->next->data==x)

{

tmp=p->next;

p->next=tmp->next;

if(p->next!=NULL)

{

tmp->next->prev=p;

}

free(tmp);

printf("Element %d has been deleted\n", x);

return;

}

p=p->next;

}

printf("Element %d not found\n", x);

return;

}

void DeletionAtPos()

{

if(start==NULL)

{

printf("Linked list is empty\n");

return;

}

struct Node \*p, \*tmp;

int pos, i;

printf("Enter the position to be deleted: ");

scanf("%d", &pos);

if(pos==1)

{

p=start;

start=start->next;

free(p);

printf("Element deleted successfully at position %d\n", pos);

return;

}

p=start;

for(i=1; i<pos-1 && p!=NULL; i++)

{

p=p->next;

}

if(p==NULL)

{

printf("There are less than %d elements\n", pos);

}

else

{

tmp=p->next;

p->next=tmp->next;

if(p->next!=NULL)

{

tmp->next->prev=p;

}

printf("Element %d deleted from position %d\n", tmp->data, pos);

free(tmp);

}

}

void count()

{

struct Node \*p;

int count=0;

p=start;

while(p!=NULL)

{

p=p->next;

count++;

}

printf("Number of elements are %d\n", count);

}

void Search()

{

if(start==NULL)

{

printf("Linked list is empty\n");

return;

}

struct Node \*p;

int pos=1, x;

printf("Enter the element to be searched: ");

scanf("%d", &x);

p=start;

while(p!=NULL)

{

if(p->data==x)

{

printf("Element %d found at position %d\n", x, pos);

return;

}

p=p->next;

pos++;

}

printf("Element %d not found\n", x);

}

void Reversal()

{

if(start==NULL)

{

printf("Linked list is empty\n");

return;

}

struct Node \*p1, \*p2;

p1=start;

p2=p1->next;

p1->next =NULL;

p1->prev=p2;

while(p2!=NULL)

{

p2->prev=p2->next;

p2->next=p1;

p1=p2;

p2=p2->prev;

}

start=p1;

printf("List is reversed\n");

return;

}

void display()

{

struct Node \*p;

if(start==NULL)

{

printf("Linked list is empty\n");

return;

}

else

{

p=start;

printf("NULL ->");

while(p!=NULL)

{

printf(" %d -> ", p->data);

p=p->next;

}

printf("NULL\n");

}

}

int main()

{

int ch;

printf("Menu:\n");

printf("1. Insertion at beginning\n");

printf("2. Insertion at end\n");

printf("3. Insertion before a node\n");

printf("4. Insertion after a node\n");

printf("5. Insertion at a given position\n");

printf("6. Deletion at beginning\n");

printf("7. Deletion at end\n");

printf("8. Deletion in between\n");

printf("9. Insertion at a given position\n");

printf("10. Count\n");

printf("11. Search\n");

printf("12. Reversal\n");

printf("13. Display\n");

printf("14. Exit\n");

printf("\n");

while(1)

{

printf("Enter your choice: ");

scanf("%d", &ch);

switch(ch)

{

case 1:

InsertAtBeg();

break;

case 2:

InsertAtEnd();

break;

case 3:

InsertBeforeNode();

break;

case 4:

InsertAfterNode();

break;

case 5:

InsertAtPos();

break;

case 6:

DeletionAtBeg();

break;

case 7:

DeletionAtEnd();

break;

case 8:

DeletionInBet();

break;

case 9:

DeletionAtPos();

break;

case 10:

count();

break;

case 11:

Search();

break;

case 12:

Reversal();

break;

case 13:

display();

break;

case 14:

while (start != NULL)

{

struct Node\* tmp = start;

start = start->next;

free(tmp);

}

exit(0);

default:

printf("Invalid choice, please try again\n");

}

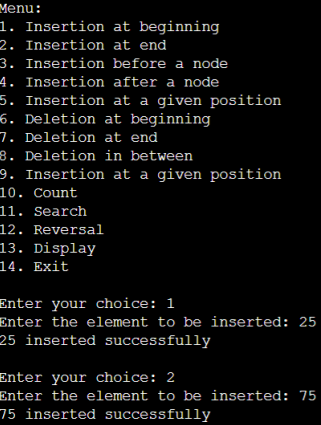
printf("\n");

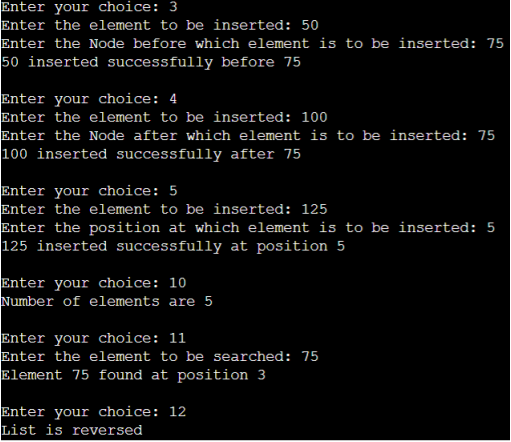
}

return 0;

}

**Output:**

****

****