**NAME:** MIRUTHULA.B

**REGISTER NO:** 231501100

**EX 1:** Implementation of single linked list

**PROGRAM:**

#include <stdlib.h>

#include <stdio.h>

struct node

{

int data;

struct node \*link;

}\*FIRST=NULL;

void Insert\_Begin(int);

void Insert\_End(int);

void Insert\_Betwn(int,int);

void Delete\_data(int);

void Delete\_pos(int);

int count();

void display();

int IsEmpty();

int IsLast();

int Search(int);

int FindNext(int);

int FindPrev(int);

int FindPrev(int v){

int pos=Search(v);

struct node \*temp;

temp=FIRST;

if (!(pos==0)){

for (int i=0;i<pos-1;i++)

{

temp=temp->link;

}

int Dat=temp->data;

return Dat;

}

else

return -1;

}

int FindNext(int v){

int pos=Search(v);

struct node \*temp;

temp=FIRST;

if (!IsLast(v) && (pos!=-1)){

for (int i=0;i<pos+1;i++)

{

temp=temp->link;

}

int Dat=temp->data;

return Dat;

}

else

return -1;

}

int Search(int value){

struct node \*temp;

temp=FIRST;

int count=0,op,flag=0;

while (temp!=NULL)

{

if (temp->data==value)

{

op=count;

flag=1;

break;

}

else

{

temp=temp->link;

}

count++;

}

if (flag)

return count;

else

return -1;

}

int IsLast(int val){

struct node \*temp;

temp=FIRST;

while (temp->link!=NULL)

{

temp=temp->link;

}

if (temp->data==val)

return 1;

else

return 0;

}

int IsEmpty(){

if (FIRST==NULL)

return 1;

else

return 0;

}

void Insert\_Begin(int dat)

{

struct node \*newnode;

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

newnode->data=dat;

if (FIRST==NULL)

{

newnode->link=NULL;

}

else

{

newnode->link=FIRST;

}

FIRST=newnode;

}

void Insert\_End(int dat)

{

struct node \*newnode,\*temp;

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

newnode->data=dat;

if (FIRST==NULL)

{

newnode->link=NULL;

FIRST=newnode;

}

else

{

temp=FIRST;

while (temp->link!=NULL)

{

temp=temp->link;

}

temp->link=newnode;

newnode->link=NULL;

}

}

void Insert\_Betwn(int dat,int pos)

{

struct node \*newnode,\*temp,\*TEMP;

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

newnode->data=dat;

int countlist;

countlist=count();

TEMP=FIRST;

if (countlist<pos)

{

printf("INVALID POSITION");

}

else

{

temp=FIRST;

for (int i=1;i<pos;i++)

{

temp=temp->link;

}

newnode->link=temp->link;

temp->link=newnode;

}

}

int count()

{

struct node \*temp;

int count=0;

temp=FIRST;

while (temp!=NULL)

{

temp=temp->link;

count++;

}

return count;

}

void Delete\_data(int dat)

{

struct node \*temp=FIRST,\*prev=NULL;

if (temp!=NULL && temp->data==dat)

{

FIRST=temp->link;

}

while (temp!=NULL && temp->data!=dat)

{

prev=temp;

temp=temp->link;

}

if (temp==NULL)

printf("DATA NOT FOUND");

prev->link=temp->link;

}

void Delete\_pos(int pos)

{

struct node \*temp,\*TEMP,\*prev=NULL;

temp=FIRST;

if (pos==1)

FIRST=temp->link;

int count;

TEMP=FIRST;

while (TEMP!=NULL)

{

TEMP=TEMP->link;

count++;

}

if (count<pos)

{

printf("INVALID POSITION\n\n");

}

else

{

for (int i=0;i<pos-1;i++)

{

if (temp!=NULL)

{

prev=temp;

temp=temp->link;

}

else

printf("INVALID POSITION");

}

prev->link=temp->link;

}

}

void display()

{

struct node \*temp;

temp=FIRST;

while (temp!=NULL)

{

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

temp=temp->link;

}

}

int main()

{

int t=1,choice,d,p;

while (t==1)

{

printf("\n\n1.Insert a node at the begining.\n2.Insert a node at the end.\n3.Insert a node at a given position.\n4.Delete a node by data.\n5.Delete a node by position.\n6.Display.\n7.Count.\n8.Exit\n9.Check if Empty\n10.Check if Element is at last\n11.search\n12.Find next number\n13.find previous number\n14.Delete list\n");

printf("Enter your choice:");

scanf("%d",&choice);

if (choice==1)

{

printf("Enter data:");

scanf("%d",&d);

Insert\_Begin(d);

}

else if (choice==2)

{

printf("Enter data:");

scanf("%d",&d);

Insert\_End(d);

}

else if (choice==3)

{

printf("Enter data:");

scanf("%d",&d);

printf("Enter position:");

scanf("%d",&p);

Insert\_Betwn(d,p);

}

else if (choice==4)

{

printf("Enter data:");

scanf("%d",&d);

Delete\_data(d);

}

else if (choice==5)

{

printf("Enter position:");

scanf("%d",&d);

Delete\_pos(d);

}

else if (choice==6)

display();

else if (choice==7)

printf("%d",count());

else if (choice==8)

{

t=0;

break;

}

else if (choice==9)

{

if (IsEmpty())

printf("The Element is at the last position");

else

printf("The element is not at the last position");

}

else if (choice==10)

{

int value;

printf("Enter the value to be checked: ");

scanf("%d",&value);

if (IsLast(value))

printf("The Element is at the last position");

else

printf("The element is not at the last position");

}

else if (choice==11)

{

int val;

printf("Enter value to Search:");

scanf("%d",&val);

if (Search(val)!=-1)

printf("The element is found at index: %d",Search(val));

else

printf("The element is not found");

}

else if (choice==12)

{

int val;

printf("Enter value:");

scanf("%d",&val);

if (FindNext(val)!=-1)

printf("The element at the next index is: %d",FindNext(val));

else

printf("INVALID NUMBER");

}

else if (choice==13)

{

int val;

printf("Enter value:");

scanf("%d",&val);

if (FindPrev(val)!=-1)

printf("The element at the prev index is: %d",FindPrev(val));

else

printf("INVALID NUMBER");

}

else if (choice==14)

{

struct node \*temp;

temp=FIRST;

for (int i=0;i<count()-1;i++)

{

free(temp);

temp=temp->link;

}

FIRST=NULL;

}

else

printf("INVALID CHOICE");

}

return 0;

}

**OUTPUT:**

