//DOUBLY LINKED LIST

#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node\* left;

struct node\* right;

};

struct node\* head=NULL;

void first();

void last();

void display();

void search();

void deletefront();

void deletelast();

void insertpos();

void main()

{

int op;

{

do

{

printf("\n1.insert at first\n2.insert at last\n3.display\n4.search\n5.Deletefront \n6.Deletelast \n7.insertpos\neneter any option");

scanf("%d",&op);

switch(op)

{

case 1:first();

break;

case 2:last();

break;

case 3:display();

break;

case 4:search();

break;

case 5:deletefront();

break;

case 6:deletelast();

break;

case 7:insertpos();

break;

}

}

while(op!=8);

}

}

void first()

{

struct node\* ne;

int x;

{ printf("------------------\n");

printf("enter the element");

scanf("%d",&x);

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

if(ne==NULL)

{

printf("insuffient memory");

return;

}

ne->data=x;

ne->left=NULL;

ne->right=NULL;

if(head==NULL)

{

head=ne;

}

else

{

ne->right=head;

head->left=ne;

head=ne;

}

}

}

void last()

{

struct node \*ne,\*ptr;

int x;

{

printf("\nenter the element\n");

scanf("%d",&x);

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

if(ne==NULL)

{

printf("insufficient");

return;

}

ne->data=x;

ne->left=NULL;

ne->right=NULL;

if(head==NULL)

{

head=ne;

}

else

{

ptr=head;

while(ptr->right!=NULL)

{

ptr=ptr->right;

}

ptr->right=ne;

ne=ptr->left;

}

}

}

void display()

{

struct node \*ptr;

if (head==NULL)

{

printf("empty");

}

ptr=head;

while(ptr!=NULL)

{

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

ptr=ptr->right;

}

}

void search()

{

int x;

printf("\nenter the element to search\n");

scanf("%d",&x);

struct node \*ptr;

if(head==NULL)

{

printf("empty");

}

ptr=head;

while(ptr!=NULL)

{ if(ptr->data==x)

{

printf("\nelement present in list\n");

}

ptr=ptr->right;

}

if(ptr=NULL)

{

printf("\nelement is not in the list\n");

}

}

void deletefront()

{

struct node \*ptr;

if(head==NULL)

{

printf("list is empty");

}

ptr=head;

head=head->right;

if(head!=NULL)

{

head->left=NULL;

}

free(ptr);

}

void deletelast()

{

struct node\* ptr, \*prev;

if (head==NULL)

{

printf("list is empty");

}

if(head->right==NULL)

{

free(head);

head=NULL;

}

ptr=head;

while(ptr->right!=NULL)

{

ptr=ptr->right;

}

prev=ptr->left;

prev->right=NULL;

free(ptr);

}

void insertpos()

{

struct node \*ne,\*ptr,\*ptr1;

int x,n;

printf("enter the element");

scanf("%d",&x);

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

if(ne==NULL)

{

printf("insufficient memory");

}

ne->data=x;

ne->right=NULL;

ne->left=NULL;

if(head==NULL)

{

head=ne;

}

ptr=head;

while(ptr->right!=NULL && ptr->data!=n)

{

ptr=ptr->right;

}

if(ptr->right==NULL)

{

ptr->right=ne;

ne->left=ptr;

}

else

ptr1=ptr->right;

ne->right=ptr1;

ne->left=ptr;

ptr->right=ne;

ptr1->left=ne;

}



