/\*linkedlist traversal,insertion,deletion,searching\*/

#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*next;

};

struct node\*head;

void beginsert();

void lastinsert();

void randominsert();

void begin\_delete();

void last\_delete();

void random\_delete();

void search();

void display();

int main()

{

int choice=0;

while(choice!=9)

{

printf("\*\*main menu\*\*\n");

printf("choose one option from the following list...\n");

printf("1.insert in begining\n2.insert at last\n3.insert at any random location\n4.delete from the begining\n5.delete from the last\n6.delete node after specefide location\n7.search for an element\n8.show\n9.exit\n");

printf("enter your choice\n");

scanf("%d",&choice);

switch(choice)

{

case 1:beginsert();

break;

case 2:lastinsert();

break;

case 3:randominsert();

break;

case 4:begin\_delete();

break;

case 5:last\_delete();

break;

case 6:random\_delete();

break;

case 7:search();

break;

case 8:display();

break;

case 9:exit(0);

break;

default:

printf("invalid choice\n");

}

}

}

void beginsert()

{

struct node\*ptr;

int item;

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

if(ptr==NULL)

{

printf("OVERFLOW\n");

}

else

{

printf("enter value\n");

scanf("%d",&item);

ptr->data=item;

ptr->next=head;

head=ptr;

printf("node inserted\n");

}

}

void lastinsert()

{

struct node\*ptr,\*temp;

int item;

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

if(ptr==NULL)

{

printf("OVERFLOW\n");

}

else

{

printf("enter value\n");

scanf("%d",&item);

ptr->data=item;

if(head==NULL) //for one node

{

ptr->next=NULL;

head=ptr;

printf("node inserted\n");

}

else

{

temp=head; //for many nodes

while(temp->next!=NULL)

{

temp=temp->next;

}

temp->next=ptr;

ptr->next=NULL;

printf("node inserted\n");

}

}

}

void randominsert()

{

int i,loc,item;

struct node\*ptr,\*temp;

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

if(ptr==NULL)

{

printf("OVERFLOW\n");

}

else

{

printf("enter element value\n");

scanf("%d",&item);

ptr->data=item;

printf("enter the location after which you want to insert\n");

scanf("%d",&loc);

temp=head;

for(i=0;i<loc;i++)

{

temp=temp->next;

if(temp==NULL)

{

printf("can not insert\n");

return ;

}

}

ptr->next=temp->next;

temp->next=ptr;

printf("node inserted\n");

}

}

void begin\_delete()

{

struct node\*ptr;

if(head==NULL)

{

printf("list is empty\n");

}

else

{

ptr=head;

head=ptr->next;

free(ptr);

printf("node deleted from the begining\n");

}

}

void last\_delete()

{

struct node\*ptr;

struct node\*ptr1;

if(head==NULL)

{

printf("list is empty\n");

}

else if(head->next==NULL)

{

head=NULL;

free(head);

printf("only node of the list deleted\n");

}

else

{

ptr=head;

while(ptr->next!=NULL)

{

ptr1=ptr;

ptr=ptr->next;

}

ptr1->next=NULL;

free(ptr);

printf("deleted node from the last...\n");

}

}

void random\_delete()

{

struct node\*ptr,\*ptr1;

int loc,i;

printf("enter the location of the node after which you want to perform deletion\n");

scanf("%d",&loc);

ptr=head;

for(i=0;i<loc;i++)

{

ptr1=ptr;

ptr=ptr->next;

if(ptr==NULL)

{

printf("can not delete\n");

return;

}

}

ptr1->next=ptr->next;

free(ptr);

printf("deleted node %d",loc+1);

}

void search()

{

struct node\*ptr;

int item,i=0,flag=0,loc;

ptr=head;

if(ptr==NULL)

{

printf("empty list\n");

}

else

{

printf("enter item which you want to search\n");

scanf("%d",&item);

while(ptr!=NULL)

{

if(ptr->data==item)

{

flag=1;

loc=i+1;

break;

}

else

{

flag=0;

}

++i;

ptr=ptr->next;

}

if(flag==0)

{

printf("item not found\n");

}

else

{

printf("item found at location %d\n",loc);

}

}

}

void display() //traversal

{

struct node\*ptr;

ptr=head;

if(ptr==NULL)

{

printf("nothing to print\n");

}

else

{

printf("printing values...\n");

while(ptr!=NULL)

{

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

ptr=ptr->next;

}

}

}

