/\*circular ll full op\*/

#include<stdio.h>

#include<stdlib.h>

struct node{

int data;

struct node\*link;

};

struct node\*header;

struct node\*create\_cll(struct node\*);

struct node\*display(struct node\*);

struct node\*insert\_beg(struct node\*);

struct node\*insert\_end(struct node\*);

struct node\*insert\_any(struct node\*);

struct node\*delete\_beg(struct node\*);

struct node\*delete\_end(struct node\*);

struct node\*delete\_any(struct node\*);

void search();

struct node\*sort\_list(struct node\*);

int main(){

int ch;

while(ch!=11){

printf("MAIN MENU\n");

printf("1.create list\n2.display\n3.insert\_beg\n4.insert\_end\n5.insert\_any\n6.delete\_beg\n7.delete\_end\n8.delete\_any\n9.search\n10.sort\_list\n11.exit\n");

printf("enter your choice: \n");

scanf("%d",&ch);

switch(ch){

case 1:header=create\_cll(header);

break;

case 2:header=display(header);

break;

case 3:header=insert\_beg(header);

break;

case 4:header=insert\_end(header);

break;

case 5:header=insert\_any(header);

break;

case 6:header=delete\_beg(header);

break;

case 7:header=delete\_end(header);

break;

case 8:header=delete\_any(header);

break;

case 9:search();

break;

case 10:header=sort\_list(header);

break;

case 11:exit(0);

default:

printf("invalid choice\n");

}

}

}

struct node\*create\_cll(struct node\*header){

int item;

struct node\*new\_node,\*ptr;

printf("enter -1 to end\n");

printf("enter the data: \n");

scanf("%d",&item);

while(item!=-1){

new\_node=(struct node\*)malloc(sizeof(struct node));

new\_node->data=item;

if(header==NULL){

new\_node->link=new\_node;

header=new\_node;

}

else{

ptr=header;

while(ptr->link!=header){

ptr=ptr->link;

}

new\_node->link=header;

ptr->link=new\_node;

}

printf("enter the data: \n");

scanf("%d",&item);

}

printf("list created\n");

return header;

}

struct node\*display(struct node\*header){

printf("the list is below\n");

struct node\*ptr;

ptr=header;

while(ptr->link!=header){

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

ptr=ptr->link;

}

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

return header;

}

struct node\*insert\_beg(struct node\*header){

struct node\*new\_node,\*ptr;

int item;

if(header==NULL){

printf("overflow,insertion not possible\n");

}

else{

printf("enter the data to be inserted: \n");

scanf("%d",&item);

new\_node=(struct node\*)malloc(sizeof(struct node\*));

new\_node->data=item;

ptr=header;

while(ptr->link!=header){

ptr=ptr->link;

}

new\_node->link=header;

ptr->link=new\_node;

header=new\_node;

}

printf("node inserted\n");

return header;

}

struct node\*insert\_end(struct node\*header){

struct node\*ptr,\*new\_node;

int item;

if(header==NULL){

printf("overflow,insertion not possible\n");

}

else{

printf("enter the data to be inserted: \n");

scanf("%d",&item);

new\_node=(struct node\*)malloc(sizeof(struct node\*));

new\_node->data=item;

ptr=header;

while(ptr->link!=header){

ptr=ptr->link;

}

new\_node->link=header;

ptr->link=new\_node;

}

printf("node inserted\n");

return header;

}

struct node\*insert\_any(struct node\*header){

struct node\*new\_node,\*ptr;

int item,loc,i;

if(header==NULL){

printf("overflow,insertion not possible\n");

}

else{

printf("enter the location at which the data has to be inserted: \n");

scanf("%d",&loc);

printf("enter the data to be inserted: \n");

scanf("%d",&item);

new\_node=(struct node\*)malloc(sizeof(struct node));

new\_node->data=item;

ptr=header;

for(i=0;i<loc-1;i++){

ptr=ptr->link;

}

new\_node->link=ptr->link;

ptr->link=new\_node;

}

printf("node inserted at specific position\n");

return header;

}

struct node\*delete\_beg(struct node\*header){

struct node\*ptr;

if(header==NULL)

{

printf("deletion not possible\n");

}

else{

ptr=header;

while(ptr->link!=header){

ptr=ptr->link;

}

ptr->link=header->link;

free(header);

header=ptr->link;

}

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

return header;

}

struct node\*delete\_end(struct node\*header){

struct node\*ptr,\*ptr1;

if(header==NULL)

{

printf("deletion not possible\n");

}

else{

ptr=header;

while(ptr->link!=header){

ptr1=ptr;

ptr=ptr->link;

}

ptr1->link=header;

free(ptr);

}

printf("node is deleted from the end\n");

return header;

}

struct node\*delete\_any(struct node\*header){

struct node\*ptr,\*ptr1;

int item,loc,i;

if(header==NULL)

{

printf("deletion not possible\n");

}

else{

ptr=header;

printf("enter the location after which the node has to be deleted\n");

scanf("%d",&loc);

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

ptr1=ptr;

ptr=ptr->link;

}

ptr1->

link=ptr->link;

free(ptr);

}

printf("node deleged from specific position\n");

return header;

}

void search()

{

struct node\*ptr;

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

if(header==NULL)

{

printf("list is empty\n");

}

else

{

printf("enter the data to be searched: \n");

scanf("%d",&item);

ptr=header;

while(ptr!=NULL)

{

if(ptr->data==item)

{

flag=1;

loc=i+1;

break;

}

else

{

flag=0;

}

++i;

ptr=ptr->link;

}

if(flag==0)

{

printf("search item not found\n");

}

else

{

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

}

}

}

struct node\*sort\_list(struct node\*header)

{

struct node\*ptr1,\*ptr2;

int temp;

ptr1=header;

while(ptr1->link!=header)

{

ptr2=ptr1->link;

while(ptr2!=header) //there are atleast 2 nodes in the list

{

if(ptr1->data>ptr2->data)

{

temp=ptr1->data;

ptr1->data=ptr2->data;

ptr2->data=temp;

}

ptr2=ptr2->link;

}

ptr1=ptr1->link;

}

printf("list sorted\n");

return header;

}

