//single linked lists - insert,delete,search operations

#include<stdio.h>

#include<stdlib.h>

struct node{

int data;

struct node \*next;

};

struct node \*head;

int count =0;

void display(){

if(head==NULL){//if count is 0

printf("Empty list.\n");

}

else{

struct node \*p;

p=head;

while(p->next!=NULL){

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

p=p->next;

}

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

}

}

void insert\_begin(){

int value;

printf("Enter value to insert at beginning: ");

scanf("%d",&value);

struct node \*temp;

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

if(temp==NULL){

printf("Memory insufficient for node creation");

}else{

temp->data= value;

if(head==NULL){

temp->next=NULL;

}else{

temp->next=head;

}

head=temp;

count++;

}

}

void insert\_end(){

int value;

printf("Enter value to insert at end: ");

scanf("%d",&value);

struct node \*temp;

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

if(temp==NULL){

printf("Memory insufficient for node creation");

}else{

temp->data= value;

temp->next=NULL;

if(head==NULL){//if it is first node in list

head=temp;

}else{

struct node \*p;

p=head;

while(p->next!=NULL){//move till end of list

p=p->next;

}

p->next=temp;//adding element to list

}

count++;//increasing the count

}

}

void insert\_pos(){

int value,pos;

printf("Enter desired position to enter: ");

scanf("%d",&pos);

if(pos>0&& pos<=count+1){

struct node \*temp;

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

if(temp==NULL){

printf("Memory insufficient for node creation");

}else{

if(pos==1)

insert\_begin();

else if(pos==count+1)

insert\_end();

else{

printf("Enter value to insert: ");

scanf("%d",&value);

struct node \*p;

p=head;

for(int i=1;i<=pos-2;i++){

p=p->next;

}

temp->data=value;

temp->next=p->next;

p->next=temp;

count++;

}

}

}

else

printf("Position given is out of bounds\n");

}

void del\_beg(){

if(head==NULL)

printf("Cannot delete as list is empty");

else{

struct node \*p;

p=head;

head=head->next;

free(p);

count--;

printf("First node deleted successfully!");

}

}

void del\_end(){

if(head==NULL)

printf("Cannot delete as list is empty");

else{

struct node \*p,\*q;

p=head;

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

p=p->next;

}

q=p;

p=p->next;

q->next=NULL;

free(p);

count--;

printf("Last node is deleted successfully!");

}

}

void del\_pos(){

int pos;

printf("Enter desired position to delete: ");

scanf("%d",&pos);

if(pos>=1&& pos<=count){

if(pos==1)

del\_beg();

else if(pos==count)

del\_end();

else{

struct node \*p,\*q;

p=head;

for(int i=1;i<=pos-2;i++)

p=p->next;

q=p;

p=p->next;

q->next=p->next;

free(p);

count--;

printf("Deleted successfully!");

}

}

else

printf("Position given is out of bounds\n");

}

void search(){

int value,loc=1;

printf("Enter value to find: ");

scanf("%d",&value);

struct node \*p;

p=head;

int found=-1;

while(p->next!=NULL){

if(p->data==value){

printf("Data found at position %d",loc);

found=1;

break;

}

loc++;

p=p->next;

}

if(p->data==value && found==-1){

printf("Data found at position %d",loc);

found=1;

}

if(found==-1)

printf("Data is not present");

}

int main(){

int choice;

while(1){

printf("Option 1- Display contents\n");

printf("Option 2- Count number of nodes in list\n");

printf("Option 3- Insert at beginning\n");

printf("Option 4- Insert at end\n");

printf("Option 5- Insert at desired position\n");

printf("Option 6- Delete at beginning\n");

printf("Option 7- Delete at end\n");

printf("Option 8- Delete at desired position\n");

printf("Option 9- Search for particular element\n");

printf("\nChoose your option(1-9)[0 to exit]: ");

scanf("%d",&choice);

switch(choice){

case 1: display();

break;

case 2: printf("Count of the nodes in list: %d\n",count);

break;

case 3: insert\_begin();

break;

case 4: insert\_end();

break;

case 5: insert\_pos();

break;

case 6: del\_beg();

break;

case 7: del\_end();

break;

case 8: del\_pos();

break;

case 9: search();

break;

default: exit(0);

}

printf("\n\n");

}

return 0;

}

//double linked lists - operations

#include<stdio.h>

#include<stdlib.h>

struct node{

int data;

struct node \*nn;

struct node \*pn;

};

struct node \*head;

int count = 0;

void display(){

if(head->pn==NULL){//if count is 0

printf("Empty list.\n");

}

else{

struct node \*p;

p=head;

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

while(p->nn!=NULL){

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

p=p->nn;

}

}

}

void insert\_begin(){

int value;

printf("Enter value to insert at beginning: ");

scanf("%d",&value);

struct node \*temp;

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

if(temp==NULL){

printf("Memory insufficient for node creation");

}else{

temp->data=value;

temp->pn==NULL;

if(head==NULL){

temp->nn=NULL;

}

else{

temp->nn=head;

head=temp;

}

count++;

}

}

void insert\_end(){

int value;

printf("Enter value to insert at end: ");

scanf("%d",&value);

struct node \*temp;

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

if(temp==NULL){

printf("Memory insufficient for node creation");

}else{

temp->data= value;

temp->nn=NULL;

struct node \*p;

p=head;

while(p->nn!=NULL){

p=p->nn;

}

p->nn=temp;

temp->pn=p;

}

}

int insert\_pos(){

int value,pos;

printf("Enter desired position to enter: ");

scanf("%d",&pos);

if(pos>0&& pos<=count+1){

struct node \*temp;

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

if(temp==NULL){

printf("Memory insufficient for node creation");

}

else{

if(pos==1)

insert\_begin();

else if(pos==count+1)

insert\_end();

else{

printf("Enter value to insert: ");

scanf("%d",&value);

struct node \*p;

p=head;

for(int i=1;i<=pos-2;i++){

p=p->nn;

}

temp->pn=p;

temp->nn=p->nn;

p->nn=temp;

temp->nn->pn=temp;

count++;

}

}

}

else

printf("Position given is out of bounds\n");

}

int del\_beg(){

if(head==NULL)

printf("Cannot delete as list is empty");

if(head->)

else{

struct node \*p;

p=head;

head=head->nn;

free(p);

count--;

printf("First node deleted successfully!");

}

}

int main(){

int choice;

while(1){

printf("Option 1- Display contents\n");

printf("Option 2- Count number of nodes in list\n");

printf("Option 3- Insert at beginning\n");

printf("Option 4- Insert at end\n");

printf("Option 5- Insert at desired position\n");

printf("Option 6- Delete at beginning\n");

printf("Option 7- Delete at end\n");

printf("Option 8- Delete at desired position\n");

printf("Option 9- Search for particular element\n");

printf("Option 10- Display data in reverse order\n");

printf("\nChoose your option(1-10)[0 to exit]: ");

scanf("%d",&choice);

switch(choice){

case 1: display();

break;

case 2: printf("Count of the nodes in list: %d\n",count);

break;

case 3: insert\_begin();

break;

case 4: insert\_end();

break;

case 5: insert\_pos();

break;

case 6: del\_beg();

break;

case 7: del\_end();

break;

case 8: del\_pos();

break;

case 9: search();

break;

case 10: reverse();

break;

default: exit(0);

}

printf("\n\n");

}

return 0;

}

//double linked lists - operations

#include<stdio.h>

#include<stdlib.h>

struct node{

int data;

struct node \*nn;

struct node \*pn;

};

struct node \*head;

int count = 0;

void display(){

if(head==NULL){//if count is 0

printf("Empty list.\n");

}

else{

struct node \*p;

p=head;

while(p!=NULL){

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

p=p->nn;

}

}

}

void insert\_begin(){

int value;

printf("Enter value to insert at beginning: ");

scanf("%d",&value);

struct node \*temp;

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

if(temp==NULL){

printf("Memory insufficient for node creation");

}else{

temp->data=value;

temp->pn==NULL;

if(head==NULL){

temp->nn=NULL;

head=temp;

}

else{

temp->nn=head;

head=temp;

}

count++;

}

}

void insert\_end(){

int value;

printf("Enter value to insert at end: ");

scanf("%d",&value);

struct node \*temp;

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

if(temp==NULL){

printf("Memory insufficient for node creation");

}else{

temp->data=value;

temp->nn=NULL;

if(head==NULL){

temp->pn=NULL;

head=temp;

}else{

struct node \*p;

p=head;

while(p->nn!=NULL){

p=p->nn;

}

p->nn=temp;

temp->pn=p;

}

count++;

}

}

void insert\_pos(){

int pos;

printf("Enter desired position to enter: ");

scanf("%d",&pos);

if(pos>0&& pos<=count+1){

struct node \*temp;

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

if(temp==NULL){

printf("Memory insufficient for node creation");

}

else{

if(pos==1)

insert\_begin();

else if(pos==count+1)

insert\_end();

else{

int value;

printf("Enter value to insert: ");

scanf("%d",&value);

struct node \*p;

p=head;

for(int i=1;i<=pos-2;i++){

p=p->nn;

}

temp->pn=p;

temp->nn=p->nn;

p->nn=temp;

temp->nn->pn=temp;

temp->data=value;

count++;

}

}

}

else

printf("Position given is out of bounds\n");

}

void del\_beg(){

if(head==NULL)

printf("Cannot delete as list is empty");

else{

struct node \*p;

p=head;

head=head->nn;

free(p);

count--;

printf("First node deleted successfully!");

}

}

void del\_end(){

if(head==NULL)

printf("Cannot delete as list is empty");

else if(head->pn==NULL && head->nn==NULL)

free(head);

else{

struct node \*p;

p=head;

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

p=p->nn;

free(p->nn);

p->nn=NULL;

count--;

printf("Last node deleted successfully!");

}

}

void del\_pos(){

int pos;

printf("Enter desired position to delete: ");

scanf("%d",&pos);

if(pos>=1&& pos<=count){

if(pos==1)

del\_beg();

else if(pos==count)

del\_end();

else{

struct node \*p;

p=head;

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

p=p->nn;

p->pn->nn=p->nn;

p->nn->pn=p->pn;

free(p);

count--;

printf("Deleted successfully!");

}

}

else

printf("Position given is out of bounds\n");

}

void search(){

int value,loc=1;

printf("Enter value to find: ");

scanf("%d",&value);

struct node \*p;

p=head;

int found=-1;

while(p->nn!=NULL){

if(p->data==value){

printf("Data found at position %d",loc);

found=1;

break;

}

loc++;

p=p->nn;

}

if(p->data==value && found==-1){

printf("Data found at position %d",loc);

found=1;

}

if(found==-1)

printf("Data is not present");

}

void display\_rev(){

if(head==NULL){//if count is 0

printf("Empty list.\n");

}

else{

struct node \*p;

p=head;

while(p->nn!=NULL)

p=p->nn;

while(p!= NULL){

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

p=p->pn;

}

printf("\n");

}

}

int main(){

int choice;

while(1){

printf("Option 1- Display contents\n");

printf("Option 2- Count number of nodes in list\n");

printf("Option 3- Insert at beginning\n");

printf("Option 4- Insert at end\n");

printf("Option 5- Insert at desired position\n");

printf("Option 6- Delete at beginning\n");

printf("Option 7- Delete at end\n");

printf("Option 8- Delete at desired position\n");

printf("Option 9- Search for particular element\n");

printf("Option 10- Display data in reverse order\n");

printf("\nChoose your option(1-10)[0 to exit]: ");

scanf("%d",&choice);

switch(choice){

case 1: display();

break;

case 2: printf("Count of the nodes in list: %d\n",count);

break;

case 3: insert\_begin();

break;

case 4: insert\_end();

break;

case 5: insert\_pos();

break;

case 6: del\_beg();

break;

case 7: del\_end();

break;

case 8: del\_pos();

break;

case 9: search();

break;

case 10: display\_rev();

break;

default: exit(0);

}

printf("\n\n");

}

return 0;

}

//circular linked lists - insert,delete,search operations

#include<stdio.h>

#include<stdlib.h>

struct node{

int data;

struct node \*next;

};

struct node \*head;

int count =0;

void display(){

if(head==NULL){//if count is 0

printf("Empty list.\n");

}

else{

struct node \*p;

p=head;

while(p->next!=head){

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

p=p->next;

}

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

}

}

void insert\_begin(){

int value;

printf("Enter value to insert at beginning: ");

scanf("%d",&value);

struct node \*temp;

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

if(temp==NULL){

printf("Memory insufficient for node creation");

}else{

temp->data= value;

if(head==NULL){

temp->next=temp;

head=temp;

}else{

temp->next=head;

struct node \*p;

p=head;

while(p->next!=head){

p=p->next;

}

p->next=temp;

}

head=temp;

count++;

}

}

void insert\_end(){

int value;

printf("Enter value to insert at end: ");

scanf("%d",&value);

struct node \*temp;

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

if(temp==NULL){

printf("Memory insufficient for node creation");

}else{

temp->data= value;

if(head==NULL){//if it is first node in list

temp->next=temp;

head=temp;

}else{

temp->next=head;

struct node \*p;

p=head;

while(p->next!=head){//move till end of list

p=p->next;

}

p->next=temp;//adding element to list

}

count++;//increasing the count

}

}

void insert\_pos(){

int value,pos;

printf("Enter desired position to enter: ");

scanf("%d",&pos);

if(pos>0&& pos<=count+1){

struct node \*temp;

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

if(temp==NULL){

printf("Memory insufficient for node creation");

}else{

if(pos==1)

insert\_begin();

else if(pos==count+1)

insert\_end();

else{

printf("Enter value to insert: ");

scanf("%d",&value);

struct node \*p,\*q;

p=head;

q=head;

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

p=p->next;

}

while(q->next!=head){//move till end of list

q=q->next;

}

temp->next=p->next;

temp->data=value;

p->next=temp;

q->next=head;

count++;

}

}

}

else

printf("Position given is out of bounds\n");

}

void del\_beg(){

if(head==NULL)

printf("Cannot delete as list is empty");

if (head->next == head) {

free(head);

head = NULL;

printf("Last node is deleted successfully!");

}

else{

struct node \*p;

p=head;

while(p->next!=head)

p=p->next;

p->next=head->next;

free(head);

head=p->next;

count--;

printf("First node deleted successfully!");

}

}

void del\_end(){

if(head==NULL)

printf("Cannot delete as list is empty");

if (head->next == head) {

head = NULL;

free(head);

printf("Last node is deleted successfully!");

}

else{

struct node \*p,\*q;

p=head;

while(p->next!=head){

q=p;

p=p->next;

}

q->next=p->next;

free(p);

count--;

printf("Last node is deleted successfully!");

}

}

void del\_pos(){

int pos;

printf("Enter desired position to delete: ");

scanf("%d",&pos);

if(pos>=1&& pos<=count){

if(pos==1)

del\_beg();

else if(pos==count)

del\_end();

else{

struct node \*p,\*q;

p=head;

for(int i=1;i<=pos-2;i++)

p=p->next;

q=p;

p=p->next;

q->next=p->next;

free(p);

count--;

printf("Deleted successfully!");

}

}

else

printf("Position given is out of bounds\n");

}

void search(){

int value,loc=1;

printf("Enter value to find: ");

scanf("%d",&value);

struct node \*p;

p=head;

int found=-1;

while(p->next!=head){

if(p->data==value){

printf("Data found at position %d",loc);

found=1;

break;

}

loc++;

p=p->next;

}

if(p->data==value && found==-1){

printf("Data found at position %d",loc);

found=1;

}

if(found==-1)

printf("Data is not present");

}

void duplicates(){

if(head==NULL){//if count is 0

printf("Empty list.\n");

}

else{

struct node \*p,\*q;

p=head;

q=p->next;

while(q != head){

if(p->data != q->data){

p =q;

q =q->next;

}

else{

p->next = q->next;

free(q);

q = p->next;

}

q=q->next;

}

printf("Duplicates deleted!");

}

}

int main(){

int choice;

while(1){

printf("Option 1- Display contents\n");

printf("Option 2- Count number of nodes in list\n");

printf("Option 3- Insert at beginning\n");

printf("Option 4- Insert at end\n");

printf("Option 5- Insert at desired position\n");

printf("Option 6- Delete at beginning\n");

printf("Option 7- Delete at end\n");

printf("Option 8- Delete at desired position\n");

printf("Option 9- Search for particular element\n");

printf("Option 10- Delete Duplicates in list\n");

printf("\nChoose your option(1-10)[0 to exit]: ");

scanf("%d",&choice);

switch(choice){

case 1: display();

break;

case 2: printf("Count of the nodes in list: %d\n",count);

break;

case 3: insert\_begin();

break;

case 4: insert\_end();

break;

case 5: insert\_pos();

break;

case 6: del\_beg();

break;

case 7: del\_end();

break;

case 8: del\_pos();

break;

case 9: search();

break;

case 10: duplicates();

break;

default: exit(0);

}

printf("\n\n");

}

return 0;

}