/\*\* singly link list insert delete and search , counting the no of nodes problrm\*\*/

/\*

#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*next;

};

void display();

void insert\_begin();

void insert\_end();

void insert\_pos();

void delete\_end();

void delete\_begin();

void delete\_pos();

struct node \*head=NULL;

void display()

{

printf("Elements are ................\n");

struct node \*ptr;

if(head==NULL)

{

printf("List is empty\n");

return;

}

else

{

ptr=head;

while(ptr!=NULL)

{

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

ptr=ptr->next;

}

}

}

void insert\_begin()

{

struct node \*temp;

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

printf("Enter the value to be inserted\n");

scanf("%d",&temp->data);

temp->next=NULL;

if(head==NULL)

head=temp;

else{

temp->next=head;

head=temp;

}

}

void insert\_end()

{

struct node \*temp,\*ptr;

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

printf("Enter the value to be inserted\n");

scanf("%d",&temp->data);

temp->next=NULL;

if(head==NULL)

{

head=temp;

}

else

{

ptr=head;

while(ptr->next !=NULL)

{

ptr=ptr->next;

}

ptr->next=temp;

}

}

void insert\_pos()

{

int pos,i;

struct node \*temp,\*ptr;

printf("enter the position ");

scanf("%d",&pos);

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

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

scanf("%d",&temp->data);

temp->next=NULL;

if(pos==0)

{

temp->next=head;

head=temp;

}

else

{

for(i=0,ptr=head;i<pos-1;i++)

{

ptr=ptr->next;

}

temp->next=ptr->next;

ptr->next=temp;

}

}

void delete\_begin()

{

struct node \*ptr;

if(head==NULL)

{

printf("List is empty\n");

return;

}

else

{

ptr=head;

head=head->next;

printf("the element deleted from the list-%d\n",ptr->data);

free(ptr);

}

}

void delete\_end()

{

struct node \*ptr,\*temp;

if(head==NULL)

{

printf("list is empty\n");

return;

}

else

{

ptr=head;

while(ptr->next!=NULL)

{

temp=ptr;

ptr=ptr->next;

}

temp->next=NULL;

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

free(ptr);

}

}

void delete\_pos()

{

int i,pos;

struct node\*ptr,\*temp;

if(head==NULL)

{

printf("LIst is empty\n");

return;

}

else

{

printf("Enter the position to be deleteed\n");

scanf("%d",&pos);

if(pos==0)

{

ptr=head;

head=head->next;

printf("the item deleted from the list is-%d",ptr->data);

free(ptr);

}

else

{

ptr=head;

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

{

temp=ptr;

ptr=ptr->next;

if(ptr==NULL)

{

printf("\n position not found\n");

return;

}

}

temp->next=ptr->next;

printf("The value deleted is-%d",ptr->data);

free(ptr);

}

}

}

void search()

{

int key;

int count;

struct node \*ptr;

ptr=head;

printf("Enter the value to be searched\n");

scanf("%d",&key);

while(ptr!=NULL)

{

if(ptr->data==key)

{

count=1;

break;

}

else

{

count=0;

}

ptr=ptr->next;

}

if(count ==1)

printf("Element found\n");

else

printf("Element not found\n");

}

void counter()

{

int c=0;

struct node \*ptr;

ptr=head;

while(ptr!=NULL)

{

c++;

ptr=ptr->next;

}

printf(" the no of nodes=%d",c);

}

void main()

{

int choice;

while(1)

{

printf("\n 1. to insert at the beginning\n 2. to insert at the end\n 3.to insert at any position enter\n 4. to display\n 5. to delete beginning \n 6. to delete at end\n 7. delete at a given position\n 8. to search\n e 9. to count the no of nodes \n10. to exit\n");

printf("Entere your choice..........\n");

scanf("%d",&choice);

switch(choice)

{

case 1:insert\_begin();

break;

case 2:insert\_end();

break;

case 3:insert\_pos();

break;

case 4:display();

break;

case 5:delete\_begin();

break;

case 6: delete\_end();

break;

case 7:delete\_pos();

break;

case 8:search();

break;

case 9:counter();

break;

case 10:exit(0);

break;

deafult:printf("Enter the correct choice\n");

break;

}

}

}\*/

/\* IMPLEMENTATION OF STACK USIMG SINGLY LINKED LIST ...........\*/

/\*

#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*next;

};

struct node \*head=NULL;

void push()

{

struct node \*temp;

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

printf("Enter the data to be pushed\n");

scanf("%d",&temp->data);

temp->next=NULL;

if(head==NULL)

{

head=temp;

return;

}

else{

temp->next=head;

head=temp;

}

}

void pop()

{

struct node \*ptr;

ptr=head;

if(head==NULL)

{

printf("stack is empty item cant be popped\n");

return;

}

else{

ptr=head;

head=ptr->next;

printf("Element popped from the stack is-%d",ptr->data);

free(ptr);

}

}

void display()

{

if(head==NULL)

{

printf("stack is empty\n");

return;

}

else{

struct node \*ptr;

ptr=head;

while(ptr!=NULL)

{

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

ptr=ptr->next;

}

}

}

void main()

{

int choice;

while(1)

{

printf("\n 1. to push\n 2. to pop\n 3.to display\n 4. to exit\n");

scanf("%d",&choice);

switch(choice)

{

case 1: push();

break;

case 2: pop();

break;

case 3:display();

break;

case 4: exit(0);

break;

}

}

}

\*/

/\*Implementation of queue using singly linked list.............. \*/

/\*

#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*next;

};

struct node \*head=NULL;

void enqueue()

{

struct node \*temp;

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

printf("Enter the value to be inserted\n");

scanf("%d",&temp->data);

temp->next=NULL;

if(head==NULL)

{

head=temp;

return;

}

else{

struct node \*ptr;

ptr=head;

while(ptr->next!=NULL)

{

ptr=ptr->next;

}

ptr->next=temp;

}

}

void dequeue()

{

struct node \*ptr;

if(head==NULL)

{

printf("Queue is empty\n");

return;

}

else{

ptr=head;

head=ptr->next;

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

free(ptr);

}

}

void display()

{

if(head==NULL)

{

printf("Queue is empty\n");

return;

}

else

{

struct node \*ptr;

ptr=head;

while(ptr!=NULL)

{

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

ptr=ptr->next;

}

}

}

void main()

{

int choice;

while(1)

{

printf(" \n1. to enqueue\n 2. to dequeue\n 3. to display\n 4. to exit\n");

printf("Enter your choice\n");

scanf("%d",&choice);

switch(choice)

{

case 1:enqueue();

break;

case 2:dequeue();

break;

case 3:display();

break;

case 4:exit(0);

break;

}

}

}

\*/

/\*\*\*\* reversing the linked list\*\*\*/

/\*

#include<stdio.h>

#include<stdlib.h>

struct node{

int data;

struct node \*next;

};

struct node \*head=NULL;

void create()

{

struct node \*temp;

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

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

scanf("%d",&temp->data);

temp->next=NULL;

if(head==NULL)

{

head=temp;

return;

}

else{

struct node \*ptr;

ptr=head;

while(ptr->next!=NULL)

{

ptr=ptr->next;

}

ptr->next=temp;

}

}

void reverse(struct node \*\*head)

{

struct node \*prev=NULL;

struct node \*next=NULL;

struct node \*cur=\*head;

while(cur!=NULL)

{

next=cur->next;

cur->next=prev;

prev=cur;

cur=next;

}

\* head=prev;

}

void display()

{

if(head==NULL)

{

printf("Queue is empty\n");

return;

}

else

{

struct node \*ptr;

ptr=head;

while(ptr!=NULL)

{

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

ptr=ptr->next;

}

}

}

int main()

{

int choice;

while(1)

{

printf("1. to create link list \n 2. to reverse the list \n 3 . to display\n 4 to exit\n");

scanf("%d",&choice);

switch(choice)

{

case 1:create();

break;

case 2:reverse(&head);

break;

case 3:display();

break;

case 4:exit(0);

break;

}

}

return 0;

}

\*/

/\*\* concatinate the link list\*\*/

/\*

#include<stdio.h>

#include<stdlib.h>

struct node \*s1=NULL;

struct node \*s2=NULL;

struct node{

int data;

struct node \*next;

};

struct node \*create(struct node \*start)

{

struct node \*temp;

int n;

printf("Enter the no of nodes to be created\n");

scanf("%d",&n);

for(int i=0;i<n;i++)

{

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

printf("Enter the value to be added\n");

scanf("%d",&temp->data);

temp->next=NULL;

if(start==NULL)

start=temp;

//adding elements in the begginning of the list

else{

temp->next=start;

start=temp;

}

}

return start;

}

struct node \*concat(struct node \*t1,struct node \*t2)

{

struct node \*ptr;

ptr=t1;

while(ptr->next!=NULL)

{

ptr=ptr->next;

}

ptr->next=t2;

printf("List has been concatinated\n");

return s1;

}

void display(struct node \*p)

{

struct node \*ptr1;

ptr1=p;

while(ptr1!=NULL)

{

printf("%d\n",ptr1->data);

ptr1=ptr1->next;

}

}

int main()

{

int choice;

while(1)

{

printf("enter 1. to concat\n 2. to display\n 3. to exit\n");

scanf("%d",&choice);

struct node \*a;

switch(choice)

{

case 1:if(s1==NULL)

s1=create(s1);

s2=create(s2);

a= concat(s1,s2);

break;

case 2:display(a);

break;

case 3:exit(0);

break;

}

}

}

\*/

/\*Sorting a linked list(ascending order)\*/

/\*

#include<stdio.h>

#include<stdlib.h>

typedef struct node

{

int data;

struct node \*next;

}NODE;

NODE \*s1=NULL;

NODE \*s2=NULL;

NODE \*create(NODE \*start)

{

NODE \*temp,\*ptr;

int n;

printf("enter the no of nodes to be created\n");

scanf("%d",&n);

for(int i=0;i<n;i++)

{

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

printf("Enter the value to be inserted in node %d\n",(i+1));

scanf("%d",&temp->data);

temp->next=NULL;

if(start==NULL)

start=temp;

else{

ptr=start;

while(ptr->next!=NULL)

{

ptr=ptr->next;

}

ptr->next=temp;

}

}

return start;

}

NODE \*sort(NODE \*s)

{

NODE \*t1,\*t2;

int temp;

for(t1=s;t1->next!=NULL;t1=t1->next)

{

for(t2=t1->next;t2!=NULL;t2=t2->next)

{

if(t1->data>t2->data)

{

temp=t1->data;

t1->data=t2->data;

t2->data=temp;

}

}

}

return s;

}

void display(struct node \*p)

{

struct node \*ptr1;

ptr1=p;

while(ptr1!=NULL)

{

printf("%d\n",ptr1->data);

ptr1=ptr1->next;

}

}

int main()

{

int choice;

while(1)

{

printf("1.to create a list\n 2. to sort the list\n 3. to display the list\n");

scanf("%d",&choice);

switch(choice)

{

case 1:s1=create(s1);

break;

case 2:if(s1==NULL)

s1=create(s1);

s1= sort(s1);

break;

case 3:display(s1);

break;

}

}

return 0;

}

\*/

/\*\*\*\*\* sorting linked list in descending order\*/

/\*

#include<stdio.h>

#include<stdlib.h>

typedef struct node

{

int data;

struct node \*next;

}NODE;

NODE \*s1=NULL;

NODE \*s2=NULL;

NODE \*create(NODE \*start)

{

NODE \*temp,\*ptr;

int n;

printf("enter the no of nodes to be created\n");

scanf("%d",&n);

for(int i=0;i<n;i++)

{

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

printf("Enter the value to be inserted in node %d\n",(i+1));

scanf("%d",&temp->data);

temp->next=NULL;

if(start==NULL)

start=temp;

else{

ptr=start;

while(ptr->next!=NULL)

{

ptr=ptr->next;

}

ptr->next=temp;

}

}

return start;

}

NODE \*sort(NODE \*s)

{

NODE \*t1,\*t2;

int temp;

for(t1=s;t1->next!=NULL;t1=t1->next)

{

for(t2=t1->next;t2!=NULL;t2=t2->next)

{

if(t1->data<t2->data)

{

temp=t1->data;

t1->data=t2->data;

t2->data=temp;

}

}

}

return s;

}

void display(struct node \*p)

{

struct node \*ptr1;

ptr1=p;

while(ptr1!=NULL)

{

printf("%d\n",ptr1->data);

ptr1=ptr1->next;

}

}

int main()

{

int choice;

while(1)

{

printf("1.to create a list\n 2. to sort the list\n 3. to display the list\n");

scanf("%d",&choice);

switch(choice)

{

case 1:s1=create(s1);

break;

case 2:if(s1==NULL)

s1=create(s1);

s1= sort(s1);

break;

case 3:display(s1);

break;

}

}

return 0;

}

\*/

/\*\*\*MERGING TWO SORTED LINKED LISTS\*\*///

/\*

#include<stdio.h>

#include<stdlib.h>

typedef struct node

{

int data;

struct node\*next;

}NODE;

NODE \*s1=NULL;

NODE \*s2=NULL;

NODE \*s3=NULL;

NODE \*create(NODE \*start)

{

NODE \*temp,\*ptr;

int n;

printf("Enter the no of nodes to be created\n");

scanf("%d",&n);

for(int i=0;i<n;i++)

{

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

printf("Enter the value of node %d\n",(i+1));

scanf("%d",&temp->data);

if(start==NULL)

start=temp;

else{

ptr=start;

while(ptr->next!=NULL)

{

ptr=ptr->next;

}

ptr->next=temp;

}

}

return start;

}

NODE \*merge(NODE \*head1,NODE \*head2)

{

if(head1==NULL)

return head2;

else if(head2==NULL)

{

return head1;

}

NODE \*mergehead=NULL;

if(head1->data<=head2->data)

{

mergehead=head1;

head1=head1->next;

}

else{

mergehead=head2;

head2=head2->next;

}

NODE \*mergetail=mergehead;

while(head1!=NULL && head2!=NULL)

{

NODE \*temp=NULL;

if(head1->data<=head2->data)

{

temp=head1;

head1=head1->next;

}

else

{

temp=head2;

head2=head2->next;

}

mergetail->next=temp;

mergetail=temp;

}

if(head1!=NULL)

{

mergetail->next=head1;

}

else if(head2!=NULL)

{

mergetail->next=head2;

}

return mergehead;

}

void display(struct node \*p)

{

struct node \*ptr1;

ptr1=p;

while(ptr1!=NULL)

{

printf("%d\n",ptr1->data);

ptr1=ptr1->next;

}

}

int main()

{

int choice;

while(1)

{

printf("Enter 1. to create 2 sorted linked lists\n 2. to merge the link lists\n 3. to display the merged linked lists\n");

scanf("%d",&choice);

switch(choice)

{

case 1: s1=create(s1);

s2=create(s2);

break;

case 2: s3=merge(s1,s2);

break;

case 3:display(s3);

break;

}

}

return 0;

}