#include<stdio.h>

#include<malloc.h>

#include<stdlib.h>

struct node //structure definition

{

int info;

struct node \*prev;

struct node \*next;

};

typedef struct node node;

void display(node \*); //function declarations

node \*insert\_beg(node \*,int );

node \*insert\_end(node \*,int );

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

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

node \*insert\_pos(node \*,int,int);

node \*delete\_pos(node \*,int);

void count(node \*);

void search(node \*,int);

node \*reverse(node \*);

int main()

{

node \*start;

start=NULL;

int c;

while(c!=0) //menu driven program

{

printf("\n1.Insert at the begining\t2.Insert at the end\n3.Delete from the beginning\t4.Delete from the end\n5.Insert at a position\t6.Delete at position\n7.Count of the list\t8.Search item in the list\n9.Reverse the list\t11.Display list\n0.End\n");

scanf("%d",&c);

switch(c)

{

case 1:

{

int data;

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

scanf("%d",&data);

start=insert\_beg(start,data);

}

break;

case 2:

{

int data;

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

scanf("%d",&data);

start=insert\_end(start,data);

}

break;

case 3:start=delete\_beg(start);

break;

case 4:start=delete\_end(start);

break;

case 5:

{

/\* int data,pos;

printf("Enter a position and data to be inserted in the list\n");

scanf("%d %d",&pos,&data);\*/

start=insert\_pos(start,5,2);

}

break;

case 6:

{

int pos;

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

scanf("%d",&pos);

start=delete\_pos(start,pos);

}break;

case 7:count(start);break;

case 8:

{

int data;

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

scanf("%d",&data);

search(start,data);

}

break;

case 9:start=reverse(start);break;

case 11:display(start);

break;

case 0:printf("Exiting");

}

}

}

node\* insert\_beg(node \*start,int data) //function to insert at the beginning

{

node \*tmp;

tmp=(node\*)malloc(sizeof(node));

tmp->info=data;

tmp->next=NULL;

tmp->prev=NULL;

if(start==NULL) //for empty case

{

start=tmp;

printf("%d has been succesfully inserted at the beginning in the list\n",tmp->info);

return start;

}

else

{

start->prev=tmp;

tmp->next=start;

start=tmp;

printf("%d has been succesfully inserted at the beginning in the list\n",tmp->info);

return start;

}

}

void display(node \*start) //display the list

{

if(start==NULL) //empty list case

{

printf("The list is empty\n");

return;

}

else

{

node \*ptr=start;

printf("\nNULL<---");

while(ptr!=NULL)

{

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

printf("--->");

ptr=ptr->next;

}

printf("NULL\n\n");

return;

}

}

node \*insert\_end(node \*start,int data) //inserts at the end of the list

{

node \*tmp;

tmp=(node\*)malloc(sizeof(node));

tmp->info=data;

tmp->next=NULL;

tmp->prev=NULL;

if(start==NULL) //empty case

{

start=tmp;

printf("%d has been succesfully inserted at the end in the list\n",tmp->info);

return start;

}

else

{

node \*ptr=start;

while(ptr->next!=NULL)

{

ptr=ptr->next;

}

tmp->prev=ptr;

ptr->next=tmp;

printf("%d has been succesfully inserted at the end in the list\n",tmp->info);

return start;

}

}

node \*delete\_beg(node \*start) //deletes at the beginning

{

node \*tmp;

if(start==NULL) //empty case

{

printf("The list is empty\n");

return start;

}

else

{

if(start->next==NULL)

{

tmp=start;

printf("%d has been deleted from the beginning of the list\n",tmp->info);

start=NULL;

free(tmp);

return start;

}

tmp=start;

start=start->next;

start->prev=NULL;

printf("%d has been deleted from the beginning of the list\n",tmp->info);

free(tmp);

return start;

}

}

node \*delete\_end(node \*start) //deletes at the end

{

node \*tmp;

if(start==NULL) //empty case

{

printf("The list is empty\n");

return start;

}

else

{

if(start->next==NULL) //single node case

{

tmp=start;

printf("%d has been deleted at the end of the list\n",tmp->info);

start=NULL;

free(tmp);

return start;

}

node \*ptr=start;

while(ptr->next->next!=NULL)

{

ptr=ptr->next;

}

tmp=ptr->next;

ptr->next=tmp->next;

printf("%d has been deleted at the end of the list\n",tmp->info);

free(tmp);

return start;

}

}

node \*insert\_pos(node \*start,int data,int pos) //inserts at a particular position

{

node \*tmp;

tmp=(node\*)malloc(sizeof(node));

tmp->info=data;

tmp->next=NULL;

tmp->prev=NULL;

if(pos==1)

{

if(start==NULL) //empty case

{

start=tmp;

printf("%d has been inserted at position %d\n",data,pos);

return start;

}

start->prev=tmp; //at position 1

tmp->next=start;

start=tmp;

printf("%d has been inserted at position %d\n",data,pos);

return start;

}

else

{

node \*ptr=start;

int i;

for(i=1;ptr!=NULL;i++)

{

if(i==pos-1)

{

if(ptr->next!=NULL)

{

tmp->next=ptr->next;

tmp->prev=ptr;

tmp->next->prev=tmp;

ptr->next=tmp;

printf("%d has been inserted at position %d\n",data,pos);

return start;

}

ptr->next=tmp;

tmp->prev=ptr;

printf("%d has been inserted at position %d\n",data,pos);

return start;

}

}

printf("Cannot insert at %d position \n",pos);

return start;

}

}

node \*delete\_pos(node \*start,int pos) //deletes at position

{

if(start==NULL) //empty case

{

printf("The list is empty\n");

return start;

}

else

{

if(pos==1)

{

if(start->next==NULL) //first case and only node

{

node \*tmp;

tmp=start;

printf("%d has been deleted at position %d\n",tmp->info,pos);

start=NULL;

free(tmp);

return start;

}

node \*tmp=start;

start=start->next;

start->prev=NULL;

printf("%d has been deleted at position %d\n",tmp->info,pos);

free(tmp);

return start;

}

node \*ptr;

ptr=start;

int i;

for(i=1;ptr->next!=NULL;i++)

{

if(i==pos-1)

{

node \*tmp=ptr->next;

printf("%d has been deleted at position %d\n",tmp->info,pos);

ptr->next=tmp->next;

ptr->next->prev=ptr;

free(tmp);

return start;

}

ptr=ptr->next;

}

if(i==pos&&ptr->next==NULL) //last node

{

node \*tmp=ptr;

tmp->prev->next=NULL;

printf("%d has been deleted at position %d\n",tmp->info,pos);

free(tmp);

return start;

}

printf("Cannot delete at this position\n");

return start;

}

}

void count(node \*start) //count of the list

{

if(start==NULL) //empty case

{

printf("List is empty\n");

return;

}

else

{

int count=0;

node \*ptr;

ptr=start;

while(ptr!=NULL)

{

ptr=ptr->next;

count++;

}

printf("The count is %d\n",count);

return;

}

}

void search(node \*start,int data) //search node in a list

{

if(start==NULL) //empty list

{

printf("The list is empty\n");

return;

}

else

{

node \*ptr;

ptr=start;

int pos=1;

while(ptr!=NULL)

{

if(ptr->info==data)

{

printf("The element %d is found at location %d\n",data,pos);

return;

}

ptr=ptr->next;

pos++;

}

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

return;

}

}

node \*reverse(node \*start) //reverses a list

{

if(start==NULL) //empty case

{

printf("The list is empty \n");

return start;

}

else

{

node \*tmp,\*nStart;

node \*ptr;

ptr=start;

while(ptr!=NULL)

{

nStart=ptr;

tmp=ptr->prev;

ptr->prev=ptr->next;

ptr->next=tmp;

ptr=ptr->prev;

}

start=nStart;

printf("The list has been reversed\n");

return start;

}

}

OUTPUT:

1.Insert at the beginning 2.Insert at the end

3.Delete from the beginning 4.Delete from the end

5.Insert at a position 6.Delete at position

7.Count of the list 8.Search item in the list

9.Reverse the list 11.Display list

0.End

1

Enter the value to be inserted in the node

4

4 has been successfully inserted at the beginning in the list

1.Insert at the beginning 2.Insert at the end

3.Delete from the beginning 4.Delete from the end

5.Insert at a position 6.Delete at position

7.Count of the list 8.Search item in the list

9.Reverse the list 11.Display list

0.End

11

NULL<---4--->NULL

1.Insert at the beginning 2.Insert at the end

3.Delete from the beginning 4.Delete from the end

5.Insert at a position 6.Delete at position

7.Count of the list 8.Search item in the list

9.Reverse the list 11.Display list

0.End

2

Enter the value to be inserted in the node

4

4 has been successfully inserted at the end in the list

1.Insert at the beginning 2.Insert at the end

3.Delete from the beginning 4.Delete from the end

5.Insert at a position 6.Delete at position

7.Count of the list 8.Search item in the list

9.Reverse the list 11.Display list

0.End

11

NULL<---4--->4--->NULL

1.Insert at the beginning 2.Insert at the end

3.Delete from the beginning 4.Delete from the end

5.Insert at a position 6.Delete at position

7.Count of the list 8.Search item in the list

9.Reverse the list 11.Display list

0.End

5

5 has been inserted at position 2

1.Insert at the beginning 2.Insert at the end

3.Delete from the beginning 4.Delete from the end

5.Insert at a position 6.Delete at position

7.Count of the list 8.Search item in the list

9.Reverse the list 11.Display list

0.End

11

NULL<---4--->5--->4--->NULL

1.Insert at the beginning 2.Insert at the end

3.Delete from the beginning 4.Delete from the end

5.Insert at a position 6.Delete at position

7.Count of the list 8.Search item in the list

9.Reverse the list 11.Display list

0.End

9

The list has been reversed

1.Insert at the beginning 2.Insert at the end

3.Delete from the beginning 4.Delete from the end

5.Insert at a position 6.Delete at position

7.Count of the list 8.Search item in the list

9.Reverse the list 11.Display list

0.End

11

NULL<---4--->5--->4--->NULL

1.Insert at the beginning 2.Insert at the end

3.Delete from the beginning 4.Delete from the end

5.Insert at a position 6.Delete at position

7.Count of the list 8.Search item in the list

9.Reverse the list 11.Display list

0.End

1

Enter the value to be inserted in the node

7

7 has been successfully inserted at the beginning in the list

1.Insert at the beginning 2.Insert at the end

3.Delete from the beginning 4.Delete from the end

5.Insert at a position 6.Delete at position

7.Count of the list 8.Search item in the list

9.Reverse the list 11.Display list

0.End

1

Enter the value to be inserted in the node

9

9 has been successfully inserted at the beginning in the list

1.Insert at the beginning 2.Insert at the end

3.Delete from the beginning 4.Delete from the end

5.Insert at a position 6.Delete at position

7.Count of the list 8.Search item in the list

9.Reverse the list 11.Display list

0.End

11

NULL<---9--->7--->4--->5--->4--->NULL

1.Insert at the beginning 2.Insert at the end

3.Delete from the beginning 4.Delete from the end

5.Insert at a position 6.Delete at position

7.Count of the list 8.Search item in the list

9.Reverse the list 11.Display list

0.End

7

The count is 5

1.Insert at the beginning 2.Insert at the end

3.Delete from the beginning 4.Delete from the end

5.Insert at a position 6.Delete at position

7.Count of the list 8.Search item in the list

9.Reverse the list 11.Display list

0.End

8

Enter the data item to be searched

7

The element 7 is found at location 2

1.Insert at the beginning 2.Insert at the end

3.Delete from the beginning 4.Delete from the end

5.Insert at a position 6.Delete at position

7.Count of the list 8.Search item in the list

9.Reverse the list 11.Display list

0.End

6

Enter the position to be deleted

3

4 has been deleted at position 3

1.Insert at the beginning 2.Insert at the end

3.Delete from the beginning 4.Delete from the end

5.Insert at a position 6.Delete at position

7.Count of the list 8.Search item in the list

9.Reverse the list 11.Display list

0.End

11

NULL<---9--->7--->5--->4--->NULL

1.Insert at the beginning 2.Insert at the end

3.Delete from the beginning 4.Delete from the end

5.Insert at a position 6.Delete at position

7.Count of the list 8.Search item in the list

9.Reverse the list 11.Display list

0.End

4

4 has been deleted at the end of the list

1.Insert at the beginning 2.Insert at the end

3.Delete from the beginning 4.Delete from the end

5.Insert at a position 6.Delete at position

7.Count of the list 8.Search item in the list

9.Reverse the list 11.Display list

0.End

4

5 has been deleted at the end of the list

1.Insert at the beginning 2.Insert at the end

3.Delete from the beginning 4.Delete from the end

5.Insert at a position 6.Delete at position

7.Count of the list 8.Search item in the list

9.Reverse the list 11.Display list

0.End

4

7 has been deleted at the end of the list

1.Insert at the beginning 2.Insert at the end

3.Delete from the beginning 4.Delete from the end

5.Insert at a position 6.Delete at position

7.Count of the list 8.Search item in the list

9.Reverse the list 11.Display list

0.End

4

9 has been deleted at the end of the list

1.Insert at the beginning 2.Insert at the end

3.Delete from the beginning 4.Delete from the end

5.Insert at a position 6.Delete at position

7.Count of the list 8.Search item in the list

9.Reverse the list 11.Display list

0.End

4

The list is empty

1.Insert at the beginning 2.Insert at the end

3.Delete from the beginning 4.Delete from the end

5.Insert at a position 6.Delete at position

7.Count of the list 8.Search item in the list

9.Reverse the list 11.Display list

0.End

11

The list is empty

1.Insert at the beginning 2.Insert at the end

3.Delete from the beginning 4.Delete from the end

5.Insert at a position 6.Delete at position

7.Count of the list 8.Search item in the list

9.Reverse the list 11.Display list

0.End

3

The list is empty

1.Insert at the beginning 2.Insert at the end

3.Delete from the beginning 4.Delete from the end

5.Insert at a position 6.Delete at position

7.Count of the list 8.Search item in the list

9.Reverse the list 11.Display list

0.End

2

Enter the value to be inserted in the node

5

5 has been successfully inserted at the end in the list

1.Insert at the beginning 2.Insert at the end

3.Delete from the beginning 4.Delete from the end

5.Insert at a position 6.Delete at position

7.Count of the list 8.Search item in the list

9.Reverse the list 11.Display list

0.End

2

Enter the value to be inserted in the node

7

7 has been successfully inserted at the end in the list

1.Insert at the beginning 2.Insert at the end

3.Delete from the beginning 4.Delete from the end

5.Insert at a position 6.Delete at position

7.Count of the list 8.Search item in the list

9.Reverse the list 11.Display list

0.End

11

NULL<---5--->7--->NULL

1.Insert at the beginning 2.Insert at the end

3.Delete from the beginning 4.Delete from the end

5.Insert at a position 6.Delete at position

7.Count of the list 8.Search item in the list

9.Reverse the list 11.Display list

0.End

2

Enter the value to be inserted in the node

8

8 has been successfully inserted at the end in the list

1.Insert at the beginning 2.Insert at the end

3.Delete from the beginning 4.Delete from the end

5.Insert at a position 6.Delete at position

7.Count of the list 8.Search item in the list

9.Reverse the list 11.Display list

0.End

4

8 has been deleted at the end of the list

1.Insert at the beginning 2.Insert at the end

3.Delete from the beginning 4.Delete from the end

5.Insert at a position 6.Delete at position

7.Count of the list 8.Search item in the list

9.Reverse the list 11.Display list

0.End

9

The list has been reversed

1.Insert at the beginning 2.Insert at the end

3.Delete from the beginning 4.Delete from the end

5.Insert at a position 6.Delete at position

7.Count of the list 8.Search item in the list

9.Reverse the list 11.Display list

0.End

11

NULL<---7--->5--->NULL

1.Insert at the beginning 2.Insert at the end

3.Delete from the beginning 4.Delete from the end

5.Insert at a position 6.Delete at position

7.Count of the list 8.Search item in the list

9.Reverse the list 11.Display list

0.En