Program to implement the various operations in a linked list:

#include<stdio.h>

#include<malloc.h>

#include<stdlib.h>

struct node //declaring structure

{

int value;

struct node \*next;

};

typedef struct node node;

//function declaration

node\* display(node \*);

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

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

node\* del\_bg(node \*);

node\* del\_end(node \*);

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

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

node\* reverse(node \*);

node\* count(node \*);

node\* search(node \*,int);

int main()

{

node \*start;

start=NULL;

int c;

while(c!=0)

{

printf("\n\n1.Insert at the beginning\t2.Insert at the end of the linked list\t3.Display the list\n4.Delete at the beginning\t5.Delete at the end\t\t\t6.Insert at position\n7.Delete at position\t\t8.Reverse the list\t\t\t9.Count of the list\n10.Search an element\t\t11.Clear screen\n");

scanf("%d",&c);

switch(c)

{

case 1:

{

int val;

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

scanf("%d",&val);

start=insert\_bg(start,val);

}

break;

case 2:

{

int val;

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

scanf("%d",&val);

start=insert\_end(start,val);

}

break;

case 3:start=display(start);break;

case 4:start=del\_bg(start);break;

case 5:start=del\_end(start);break;

case 6:

{

int val,pos;

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

scanf("%d",&val);

printf("Enter the position of the node to be inserted\n");

scanf("%d",&pos);

start=insert\_pos(start,val,pos);

}

break;

case 7:

{

int pos;

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

scanf("%d",&pos);

start=delete\_pos(start,pos);

}

break;

case 8:

{

start=reverse(start);

}

break;

case 9:start=count(start);

break;

case 10:

{

int val;

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

scanf("%d",&val);

start=search(start,val);

}

break;

case 11:system("cls");break;

case 0:printf("Exiting");

}

}

return 0;

}

node\* display(node \*start) //displays the elements in the list

{

if(start==NULL)

{

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

return start;

}

node \*p;

p=start;

printf("The list is:\nstart---> ");

while(p!=NULL)

{

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

if(p->next!=NULL)

printf("---> ");

p=p->next;

}

printf("--->NULL\n");

return start;

}

node\* insert\_bg(node \*start,int value) //inserts elements at the start of the list

{

node \*tmp;

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

tmp->value=value;

tmp->next=NULL;

if(start==NULL) //if list has no elements

{

start=tmp;

printf("%d has be inserted at the beginning\n",value);

return start;

}

else //if list has more than equal to one element

{

tmp->next=start;

start=tmp;

printf("%d has be inserted at the beginning\n",value);

return start;

}

}

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

{

node \*tmp;

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

tmp->value=value;

tmp->next=NULL;

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

{

start=tmp;

printf("%d has be inserted at the end\n",value);

return start;

}

else //if list has more than equal to one element

{

node \*ptr;

ptr=start;

while(ptr->next!=NULL)

{

ptr=ptr->next;

}

ptr->next=tmp;

printf("%d has be inserted at the end\n",value);

return start;

}

}

node\* del\_bg(node \*start) //deletes at the start of the list

{

if(start==NULL)

{

printf("The list is empty\n"); //when the list is empty

return start;

}

else //has 1 or more elements in the list

{

node \*ptr;

ptr=start;

start=start->next;

printf("%d is popped out from the beginnning\n",ptr->value);

free(ptr);

return start;

}

}

node\* del\_end(node \*start) //deletes at the end of the list

{

if(start==NULL) //case if no elements in the list

{

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

return start;

}

if(start->next==NULL) //if only one element in the list

{

printf("%d is popped from the end\n",start->value);

start=NULL;

return start;

}

else //if more than 1 element in the list

{

node \*ptr;

ptr=start;

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

{

ptr=ptr->next;

}

node \*tmp;

tmp=ptr->next;

ptr->next=tmp->next;

printf("%d is popped from the end\n",tmp->value);

free(tmp);

return start;

}

}

node\* insert\_pos(node\* start,int value,int pos)//inserts at a position in a list

{

node \*tmp;

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

tmp->value=value;

tmp->next=NULL;

if(start==NULL&&pos!=1) //start is NULL and position >1

{

printf("No such position available\n");

return start;

}

if(pos==1) //to add at position 1

{

tmp->next=start;

start=tmp;

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

return start;

}

else //adds at position other than 1

{

node \*p;

p=start;

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

{

p=p->next;

if(p==NULL)

{

printf("No such position available\n");

return start;

}

}

tmp->next=p->next;

p->next=tmp;

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

return start;

}

}

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

{

node \*tmp;

if(start==NULL) //when the list is empty

{

printf("No elements in the list\n");

return start;

}

else

{

if(pos==1) //deletes at position 1

{

int val=start->value;

printf("%d value is popped of the list\n",val);

start=start->next;

return start;

}

node \*ptr;

ptr=start;

int c=1;

while(ptr->next!=NULL) //while loop to loop through the list elements

{

if(pos-1==c)

{

node \*tmp;

tmp=ptr->next;

ptr->next=tmp->next;

printf("%d value is popped of the list\n",tmp->value);

free(tmp);

return start;

}

c++;

ptr=ptr->next;

}

printf("No such position can be deleted\n");

return start;

}

}

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

{

node \*ptr,\*prev,\*next; //declaration of prev next and ptr pointer

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

{

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

return start;

}

prev=NULL;

ptr=start;

while(ptr!=NULL)

{

next=ptr->next;

ptr->next=prev;

prev=ptr;

ptr=next;

}

start=prev;

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

return start;

}

node\* count(node \*start) //to calculate the count of the list

{

int count=0;

if(start==NULL)

{

printf("Count is 0\n"); //if list is empty

return start;

}

else

{

node \*ptr;

ptr=start;

while(ptr!=NULL)

{

ptr=ptr->next;

count++;

}

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

return start;

}

}

node\* search(node \*start,int item)// to search an item in a list

{

if(start==NULL) //if no items in the list

{

printf("List is empty\n");

return start;

}

else

{

node \*ptr;

ptr=start;

int pos=1;

while(ptr!=NULL)

{

if(ptr->value==item)

{

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

return start;

}

ptr=ptr->next;

pos++;

}

printf("%d was not found in the list\n",item);

return start; }

}

OUTPUT:

1.Insert at the beginning 2.Insert at the end of the linked list 3.Display the list 4.Delete at the beginning 5.Delete at the end 6.Insert at position 7.Delete at position 8.Reverse the list 9.Count of the list 10.Search an element 11.Clear screen 3 The list is empty 1.Insert at the beginning 2.Insert at the end of the linked list 3.Display the list 4.Delete at the beginning 5.Delete at the end 6.Insert at position 7.Delete at position 8.Reverse the list 9.Count of the list 10.Search an element 11.Clear screen 4 The list is empty 1.Insert at the beginning 2.Insert at the end of the linked list 3.Display the list 4.Delete at the beginning 5.Delete at the end 6.Insert at position 7.Delete at position 8.Reverse the list 9.Count of the list 10.Search an element 11.Clear screen 5 The list is empty 1.Insert at the beginning 2.Insert at the end of the linked list 3.Display the list 4.Delete at the beginning 5.Delete at the end 6.Insert at position 7.Delete at position 8.Reverse the list 9.Count of the list 10.Search an element 11.Clear screen 6 Enter the value of the node to be inserted 25 Enter the position of the node to be inserted 5 No such position available 1.Insert at the beginning 2.Insert at the end of the linked list 3.Display the list 4.Delete at the beginning 5.Delete at the end 6.Insert at position 7.Delete at position 8.Reverse the list 9.Count of the list 10.Search an element 11.Clear screen 7 Enter the position to be delete 5 No elements in the list 1.Insert at the beginning 2.Insert at the end of the linked list 3.Display the list 4.Delete at the beginning 5.Delete at the end 6.Insert at position 7.Delete at position 8.Reverse the list 9.Count of the list 10.Search an element 11.Clear screen 9 Count is 0 1.Insert at the beginning 2.Insert at the end of the linked list 3.Display the list 4.Delete at the beginning 5.Delete at the end 6.Insert at position 7.Delete at position 8.Reverse the list 9.Count of the list 10.Search an element 11.Clear screen 10 Enter the value to be searched 5 List is empty 1.Insert at the beginning 2.Insert at the end of the linked list 3.Display the list 4.Delete at the beginning 5.Delete at the end 6.Insert at position 7.Delete at position 8.Reverse the list 9.Count of the list 10.Search an element 11.Clear screen 1 Enter the value to be inserted 70 70 has be inserted at the beginning 1.Insert at the beginning 2.Insert at the end of the linked list 3.Display the list 4.Delete at the beginning 5.Delete at the end 6.Insert at position 7.Delete at position 8.Reverse the list 9.Count of the list 10.Search an element 11.Clear screen 1 Enter the value to be inserted 65 65 has be inserted at the beginning 1.Insert at the beginning 2.Insert at the end of the linked list 3.Display the list 4.Delete at the beginning 5.Delete at the end 6.Insert at position 7.Delete at position 8.Reverse the list 9.Count of the list 10.Search an element 11.Clear screen 3 The list is: start---> 65---> 70--->NULL 1.Insert at the beginning 2.Insert at the end of the linked list 3.Display the list 4.Delete at the beginning 5.Delete at the end 6.Insert at position 7.Delete at position 8.Reverse the list 9.Count of the list 10.Search an element 11.Clear screen 2 Enter the value to be inserted 75 75 has be inserted at the end 1.Insert at the beginning 2.Insert at the end of the linked list 3.Display the list 4.Delete at the beginning 5.Delete at the end 6.Insert at position 7.Delete at position 8.Reverse the list 9.Count of the list 10.Search an element 11.Clear screen 2 Enter the value to be inserted 80 80 has be inserted at the end 1.Insert at the beginning 2.Insert at the end of the linked list 3.Display the list 4.Delete at the beginning 5.Delete at the end 6.Insert at position 7.Delete at position 8.Reverse the list 9.Count of the list 10.Search an element 11.Clear screen 3 The list is: start---> 65---> 70---> 75---> 80--->NULL 1.Insert at the beginning 2.Insert at the end of the linked list 3.Display the list 4.Delete at the beginning 5.Delete at the end 6.Insert at position 7.Delete at position 8.Reverse the list 9.Count of the list 10.Search an element 11.Clear screen 6 Enter the value of the node to be inserted 95 Enter the position of the node to be inserted 2 95 has been inserted at position 2 1.Insert at the beginning 2.Insert at the end of the linked list 3.Display the list 4.Delete at the beginning 5.Delete at the end 6.Insert at position 7.Delete at position 8.Reverse the list 9.Count of the list 10.Search an element 11.Clear screen 3 The list is: start---> 65---> 95---> 70---> 75---> 80--->NULL 1.Insert at the beginning 2.Insert at the end of the linked list 3.Display the list 4.Delete at the beginning 5.Delete at the end 6.Insert at position 7.Delete at position 8.Reverse the list 9.Count of the list 10.Search an element 11.Clear screen 9 Count is 5 1.Insert at the beginning 2.Insert at the end of the linked list 3.Display the list 4.Delete at the beginning 5.Delete at the end 6.Insert at position 7.Delete at position 8.Reverse the list 9.Count of the list 10.Search an element 11.Clear screen 7 Enter the position to be delete 2 95 value is popped of the list 1.Insert at the beginning 2.Insert at the end of the linked list 3.Display the list 4.Delete at the beginning 5.Delete at the end 6.Insert at position 7.Delete at position 8.Reverse the list 9.Count of the list 10.Search an element 11.Clear screen 3 The list is: start---> 65---> 70---> 75---> 80--->NULL 1.Insert at the beginning 2.Insert at the end of the linked list 3.Display the list 4.Delete at the beginning 5.Delete at the end 6.Insert at position 7.Delete at position 8.Reverse the list 9.Count of the list 10.Search an element 11.Clear screen 10 Enter the value to be searched 98 98 was not found in the list 1.Insert at the beginning 2.Insert at the end of the linked list 3.Display the list 4.Delete at the beginning 5.Delete at the end 6.Insert at position 7.Delete at position 8.Reverse the list 9.Count of the list 10.Search an element 11.Clear screen 10 Enter the value to be searched 75 The element 75 is found at location 3 1.Insert at the beginning 2.Insert at the end of the linked list 3.Display the list 4.Delete at the beginning 5.Delete at the end 6.Insert at position 7.Delete at position 8.Reverse the list 9.Count of the list 10.Search an element 11.Clear screen 3 The list is: start---> 65---> 70---> 75---> 80--->NULL 1.Insert at the beginning 2.Insert at the end of the linked list 3.Display the list 4.Delete at the beginning 5.Delete at the end 6.Insert at position 7.Delete at position 8.Reverse the list 9.Count of the list 10.Search an element 11.Clear screen 5 80 is popped from the end 1.Insert at the beginning 2.Insert at the end of the linked list 3.Display the list 4.Delete at the beginning 5.Delete at the end 6.Insert at position 7.Delete at position 8.Reverse the list 9.Count of the list 10.Search an element 11.Clear screen 4 65 is popped out from the beginnning 1.Insert at the beginning 2.Insert at the end of the linked list 3.Display the list 4.Delete at the beginning 5.Delete at the end 6.Insert at position 7.Delete at position 8.Reverse the list 9.Count of the list 10.Search an element 11.Clear screen 3 The list is: start---> 70---> 75--->NULL 1.Insert at the beginning 2.Insert at the end of the linked list 3.Display the list 4.Delete at the beginning 5.Delete at the end 6.Insert at position 7.Delete at position 8.Reverse the list 9.Count of the list 10.Search an element 11.Clear screen 5 75 is popped from the end 1.Insert at the beginning 2.Insert at the end of the linked list 3.Display the list 4.Delete at the beginning 5.Delete at the end 6.Insert at position 7.Delete at position 8.Reverse the list 9.Count of the list 10.Search an element 11.Clear screen 5 70 is popped from the end 1.Insert at the beginning 2.Insert at the end of the linked list 3.Display the list 4.Delete at the beginning 5.Delete at the end 6.Insert at position 7.Delete at position 8.Reverse the list 9.Count of the list 10.Search an element 11.Clear screen 3 The list is empty 1.Insert at the beginning 2.Insert at the end of the linked list 3.Display the list 4.Delete at the beginning 5.Delete at the end 6.Insert at position 7.Delete at position 8.Reverse the list 9.Count of the list 10.Search an element 11.Clear screen 1 Enter the value to be inserted 10 10 has be inserted at the beginning 1.Insert at the beginning 2.Insert at the end of the linked list 3.Display the list 4.Delete at the beginning 5.Delete at the end 6.Insert at position 7.Delete at position 8.Reverse the list 9.Count of the list 10.Search an element 11.Clear screen 1 Enter the value to be inserted 20 20 has be inserted at the beginning 1.Insert at the beginning 2.Insert at the end of the linked list 3.Display the list 4.Delete at the beginning 5.Delete at the end 6.Insert at position 7.Delete at position 8.Reverse the list 9.Count of the list 10.Search an element 11.Clear screen 1 Enter the value to be inserted 30 30 has be inserted at the beginning 1.Insert at the beginning 2.Insert at the end of the linked list 3.Display the list 4.Delete at the beginning 5.Delete at the end 6.Insert at position 7.Delete at position 8.Reverse the list 9.Count of the list 10.Search an element 11.Clear screen 3 The list is: start---> 30---> 20---> 10--->NULL 1.Insert at the beginning 2.Insert at the end of the linked list 3.Display the list 4.Delete at the beginning 5.Delete at the end 6.Insert at position 7.Delete at position 8.Reverse the list 9.Count of the list 10.Search an element 11.Clear screen 8 List has been reversed 1.Insert at the beginning 2.Insert at the end of the linked list 3.Display the list 4.Delete at the beginning 5.Delete at the end 6.Insert at position 7.Delete at position 8.Reverse the list 9.Count of the list 10.Search an element 11.Clear screen 3 The list is: start---> 10---> 20---> 30--->NULL 1.Insert at the beginning 2.Insert at the end of the linked list 3.Display the list 4.Delete at the beginning 5.Delete at the end 6.Insert at position 7.Delete at position 8.Reverse the list 9.Count of the list 10.Search an element 11.Clear screen 5 30 is popped from the end 1.Insert at the beginning 2.Insert at the end of the linked list 3.Display the list 4.Delete at the beginning 5.Delete at the end 6.Insert at position 7.Delete at position 8.Reverse the list 9.Count of the list 10.Search an element 11.Clear screen 3 The list is: start---> 10---> 20--->NULL