***NAME - AKRITI CHOUDHARY***

***ROLL NUMBER - 2005776***

***SUBJECT - DSA LAB***

***DATE - 31/8/2021***

***CLASS - B14***

***BRANCH - CSE***

***Question 1 : WAP to create a double linked list of n nodes and display the linked list by using suitable***

***user defined functions for create and display operations.***

#include <stdio.h>

#include <stdlib.h>

struct node

{

int data;

struct node \*prev;

struct node \*next;

} \* head, \*last;

void createList(int n);

void displayFromFirst();

int main()

{

int n, choice;

head = NULL;

last = NULL;

printf("Enter the number of nodes you want to create: ");

scanf("%d", &n);

createList(n);

printf("Displaying list in forward manner");

displayFromFirst();

return 0;

}

void createList(int n)

{

int i, data;

struct node \*newNode;

if (n >= 1)

{

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

if (head != NULL)

{

printf("Enter data of 1 node: ");

scanf("%d", &data);

head->data = data;

head->prev = NULL;

head->next = NULL;

last = head;

for (i = 2; i <= n; i++)

{

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

if (newNode != NULL)

{

printf("Enter data of %d node: ", i);

scanf("%d", &data);

newNode->data = data;

newNode->prev = last;

newNode->next = NULL;

last->next = newNode;

last = newNode;

}

else

{

printf("Unable to allocate memory.");

break;

}

}

}

else

{

printf("Unable to allocate memory");

}

}

}

void displayFromFirst()

{

struct node \*temp;

int n = 1;

if (head == NULL)

{

printf("List is empty.");

}

else

{

temp = head;

puts("");

while (temp != NULL)

{

printf("Node %d = %d\n", n, temp->data);

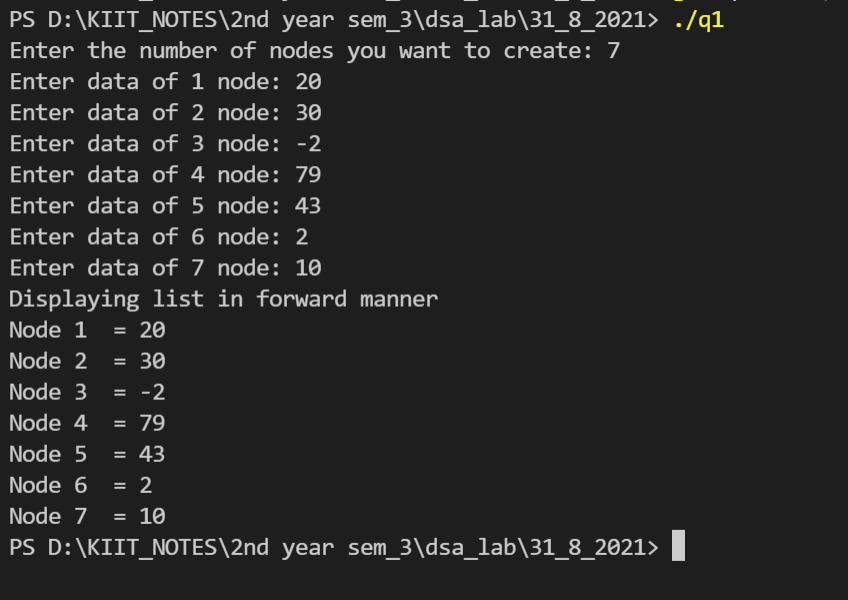
n++;

temp = temp->next;

}

}

}



***Question 2 : WAP to reverse the sequence elements in a double linked list.***

#include <stdio.h>

#include <stdlib.h>

struct node

{

int data;

struct node \*prev;

struct node \*next;

} \* head, \*last;

void createList(int n);

void displayListFromEnd();

int main()

{

int n, choice;

head = NULL;

last = NULL;

printf("Enter the number of nodes you want to create: ");

scanf("%d", &n);

createList(n);

printf("Printing list from End : ");

displayListFromEnd();

return 0;

}

void createList(int n)

{

int i, data;

struct node \*newNode;

if (n >= 1)

{

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

if (head != NULL)

{

printf("Enter data of 1 node: ");

scanf("%d", &data);

head->data = data;

head->prev = NULL;

head->next = NULL;

last = head;

for (i = 2; i <= n; i++)

{

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

if (newNode != NULL)

{

printf("Enter data of %d node: ", i);

scanf("%d", &data);

newNode->data = data;

newNode->prev = last;

newNode->next = NULL;

last->next = newNode;

last = newNode;

}

else

{

printf("Unable to allocate memory.");

break;

}

}

}

else

{

printf("Unable to allocate memory");

}

}

}

void displayListFromEnd()

{

struct node \*temp;

int n = 0;

if (last == NULL)

{

printf("List is empty.");

}

else

{

temp = last;

puts("");

while (temp != NULL)

{

printf("node %d = %d\n", n, temp->data);

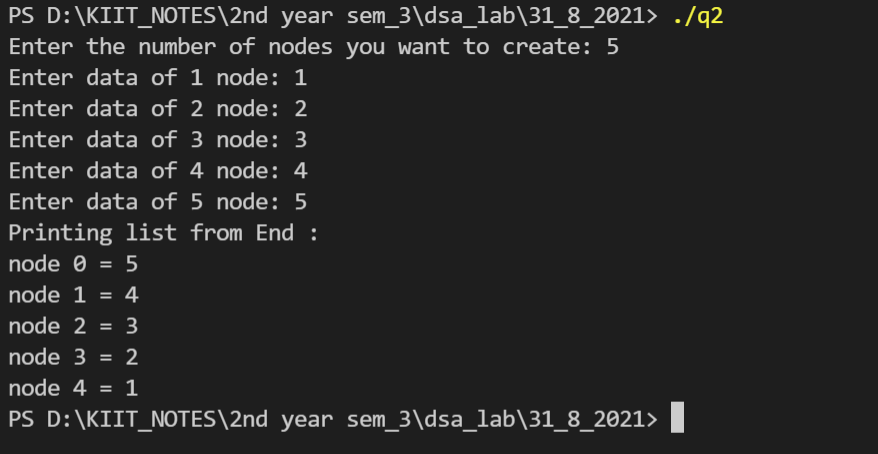
n++;

temp = temp->prev;

}

}

}



***Question 3 : Write a menu driven program to perform the following operations in a double linked list by***

***using suitable user defined functions for each case.***

***a) Traverse the list forward***

***b) Traverse the list backward***

***c) Check if the list is empty***

***d) Insert a node at the certain position (at beginning/end/any position)***

***e) Delete a node at the certain position (at beginning/end/any position)***

***f) Delete a node for the given key***

***g) Count the total number of nodes***

***h) Search for an element in the linked list***

***Verify & validate each function from main method.***

#include <stdio.h>

#include <stdlib.h>

struct node

{

int data;

struct node \* prev;

struct node \* next;

}\* head=NULL, \* tail=NULL;

void input()

{

struct node \* cur;

int n;

printf("enter the number of nodes");

scanf("%d",&n);

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

{

cur=malloc(sizeof(struct node));

printf("\nenter the %d th node",i+1);

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

if(head==NULL)

{

head=tail=cur;

head->prev=NULL;

tail->next=NULL;

}

else

{

tail->next=cur;

cur->prev=tail;

tail=cur;

tail->next=NULL;

}

}

}

void display()

{

struct node \* cur=head;

if(head==NULL)

printf("link is empty");

else

{

printf("nodes of double link list are\n");

while(cur!=NULL)

{

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

cur=cur->next;

}

}

}

void checkEmpty()

{

int c=0;

struct node \* ptr;

if(head !=NULL)

{

ptr=head;

while(ptr!=NULL)

{

c++;

ptr=ptr->next;

}

c++;

}

if(c>0)

printf("the link list is not empty\n");

else

printf("the link list is empty\n");

}

void countNode()

{

int c=0;

struct node \* ptr;

if(head !=NULL)

{

ptr=head;

while(ptr->next!=NULL)

{

c++;

ptr=ptr->next;

}

c++;

}

printf("the no. of nodes in the link list is %d\n",c);

}

void SearchNode()

{

int s;

printf("enter the element to be searched");

scanf("%d",&s);

struct node \* ptr;

ptr=head;

int flag=0;

while (ptr!= NULL)

{

if(ptr->data==s)

{

flag=1;

break;

}

else

{

ptr = ptr->next;

}

}

if(flag)

{

printf("element found\n");

}

else

{

printf("element not found\n");

}

printf("end of searching \n");

}

void insertNode()

{

int val,pos;

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

scanf("%d",&val);

printf("enter the position where the node is to be inserted");

scanf("%d",&pos);

struct node \*cur,\*ptr;

cur=malloc(sizeof(struct node));

cur->data=val;

cur->next=cur->prev=NULL;

if(head==NULL)

head=cur;

else if(pos==1)

{

cur->next=head;

head->prev=cur;

head=cur;

}

else

{

ptr=head;

int i=1;

while(i<pos-1 && ptr->next!=NULL)

{

i++;

ptr=ptr->next;

}

if(ptr->next==NULL)

{

cur->prev=ptr;

ptr->next=cur;

}

else

{

cur->prev=ptr;

cur->next=ptr->next;

cur->next->prev=cur;

ptr->next=cur;

}

}

display();

}

void deletekey()

{int val;

printf("enter the value to be deleted");

scanf("%d",&val);

struct node \*ptr;

if(head==NULL)

printf("link list is empty");

else

{

ptr=head;

while(ptr!=NULL)

{

if (ptr->data==val)

break;

ptr=ptr->next;

}

if(ptr==NULL)

printf("data not found");

else if(head==ptr && ptr->next==NULL)

{

head=NULL;

free(ptr);

}

else if(head==ptr)

{

head=head->next;

head->prev=NULL;

free(ptr);

}

else if(ptr->next==NULL)

{

ptr->prev->next=NULL;

free(ptr);

}

else

{

ptr->prev->next=ptr->next;

ptr->next->prev=ptr->prev;

free(ptr);

}

}

display();

}

void traversebackwards()

{

struct node \* temp=NULL;

struct node \* cur;

cur=head;

while(cur!=NULL)

{

temp=cur->prev;

cur->prev=cur->next;

cur->next=temp;

cur=cur->prev;

}

if(temp!=NULL)

head=temp->prev;

display();

}

void deleteNode()

{int n;

printf("enter the position where the node is to be deleted");

scanf("%d",&n);

if(head==NULL)

printf("the link list is empty");

struct node \* cur=head;

int i=1;

while(cur!=NULL && i<n)

{

cur=cur->next;

i++;

}

if(cur==NULL || head==NULL)

printf("position not in the link list");

else if(head==cur)

{

head=cur->next;

head->prev=NULL;

free(cur);

}

else if(cur->next==NULL)

{cur=tail;

tail=tail->prev;

tail->next=NULL;

free(cur);

}

else

{

cur->prev->next=cur->next;

cur->next->prev=cur->prev;

free(cur);

}

display();

}

int main()

{

int op;

input();

do

{

printf("enter 1 if u want to traverse the link list\n ");

printf("enter 2 if u want to check whether the link list is empty\n ");

printf("enter 3 if u want to insert a node in the link list\n ");

printf("enter 4 if u want to delete a node from the link list\n ");

printf("enter 5 if u want to delete a node for the given key from the link list\n ");

printf("enter 6 if u want to count the nodes in the link list\n ");

printf("enter 7 if u want to search for a node in the link list\n ");

printf("enter 8 if u want to reverse the link list\n ");

printf("enter 9 to quit\n");

printf("enter the operation to be performed in the linked list\n");

scanf("%d", &op);

switch (op)

{

case 1:

printf("operation chosen: traversing the list \n");

display();

break;

case 2:

printf("operation chosen: checking if the linked list is empty \n");

checkEmpty();

break;

case 3:

printf("operation chosen: insertion node \n");

insertNode();

break;

case 4:

printf("operation chosen: deletion node \n");

deleteNode();

break;

case 5:

printf("operarion chosen:delete a node for a given key \n");

deletekey();

break;

case 6:

printf("operation chosen: counting nodes \n");

countNode();

break;

case 7:

printf("operation chosen: searching nodes \n");

SearchNode();

break;

case 8:

printf("operation chosen: traverse backwards \n");

traversebackwards();

break;

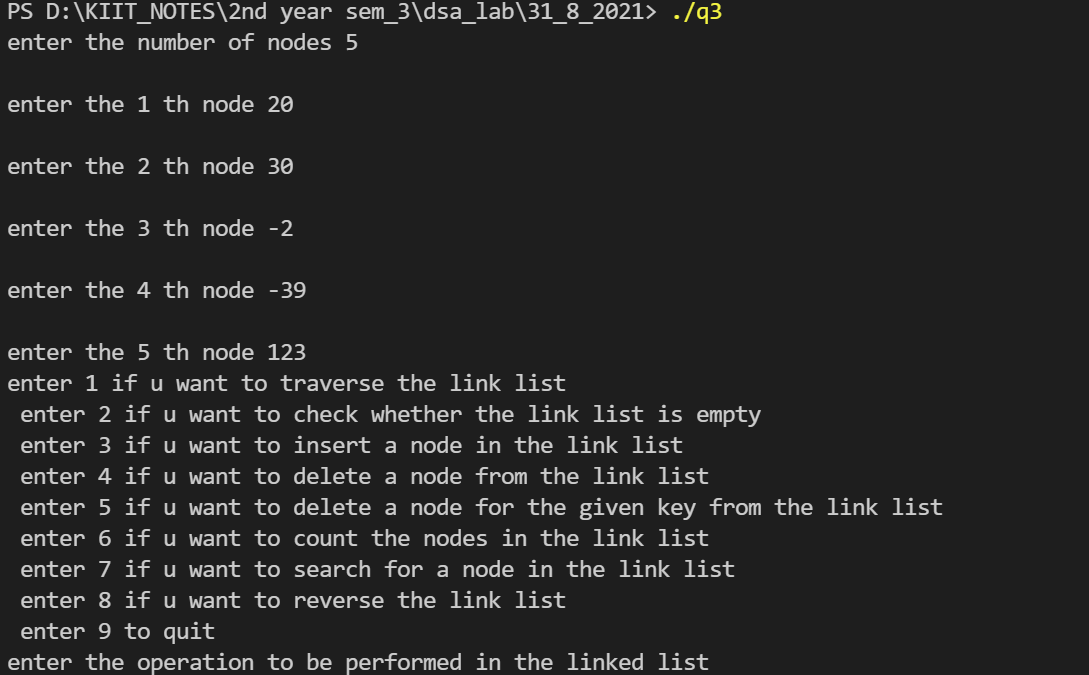
case 9:

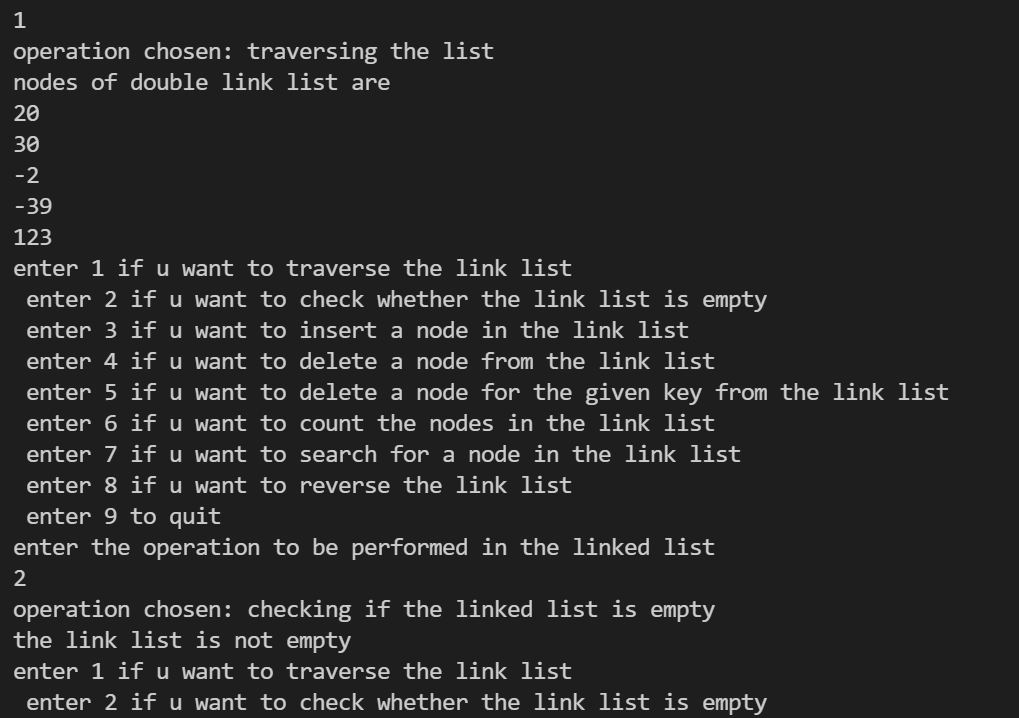
break;

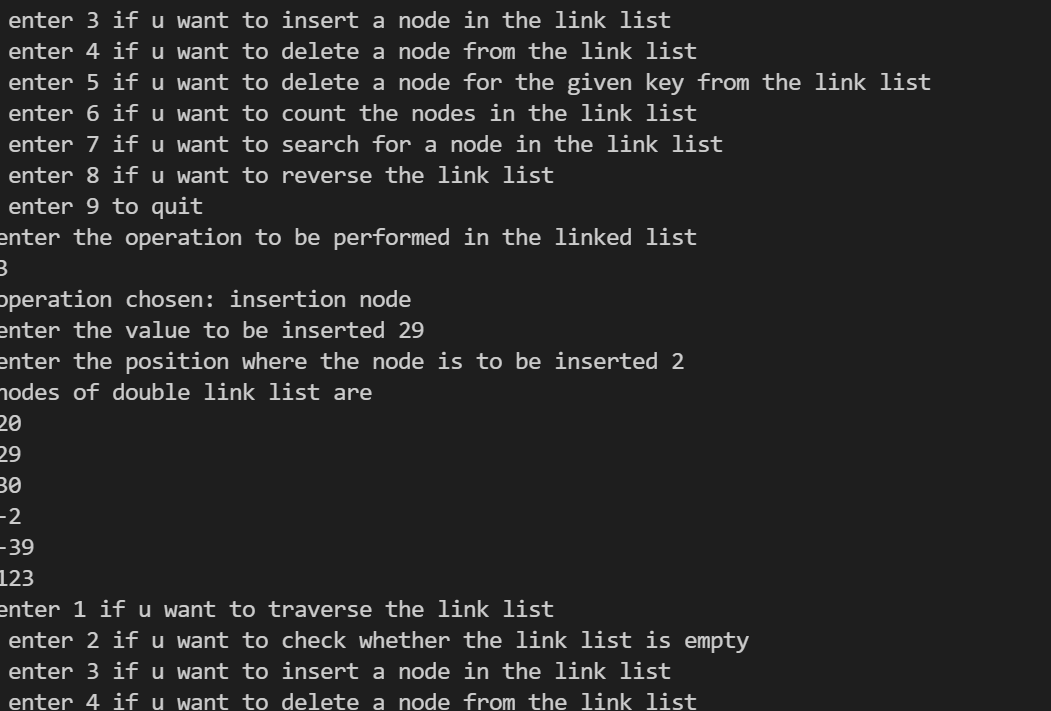
}

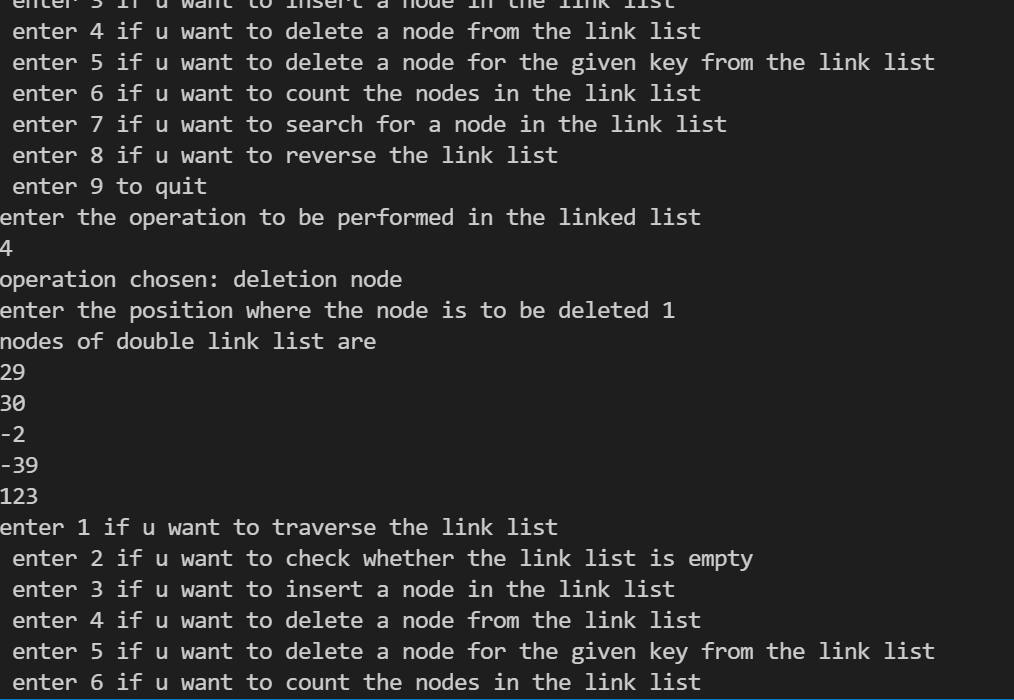
}while(op!=9);

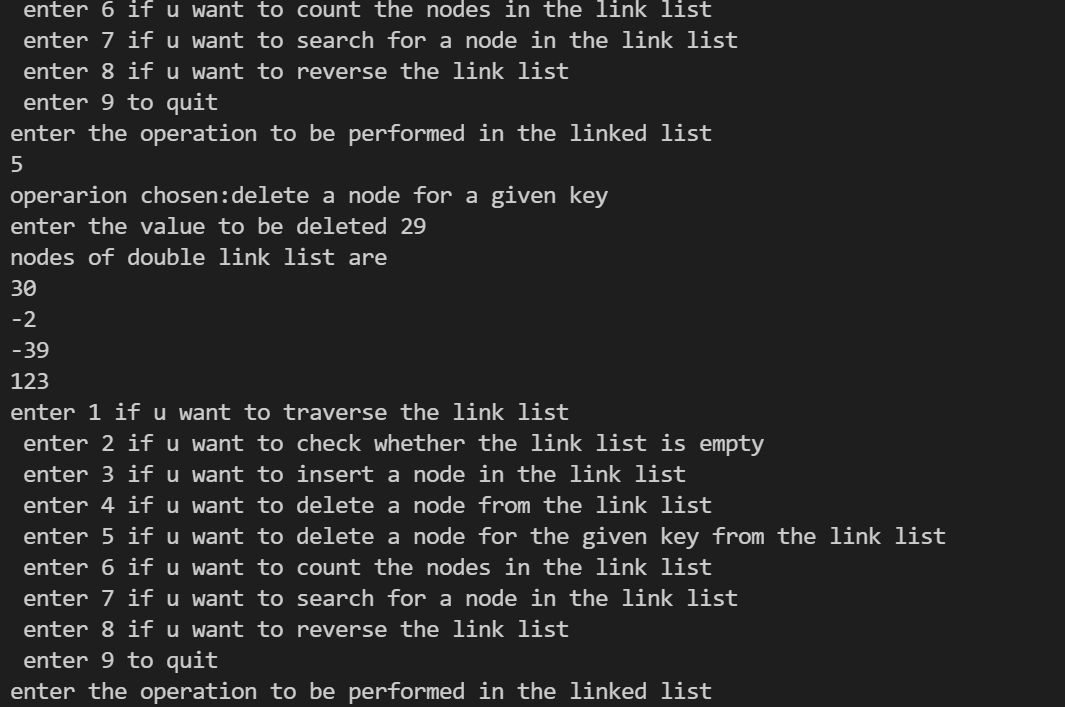
}

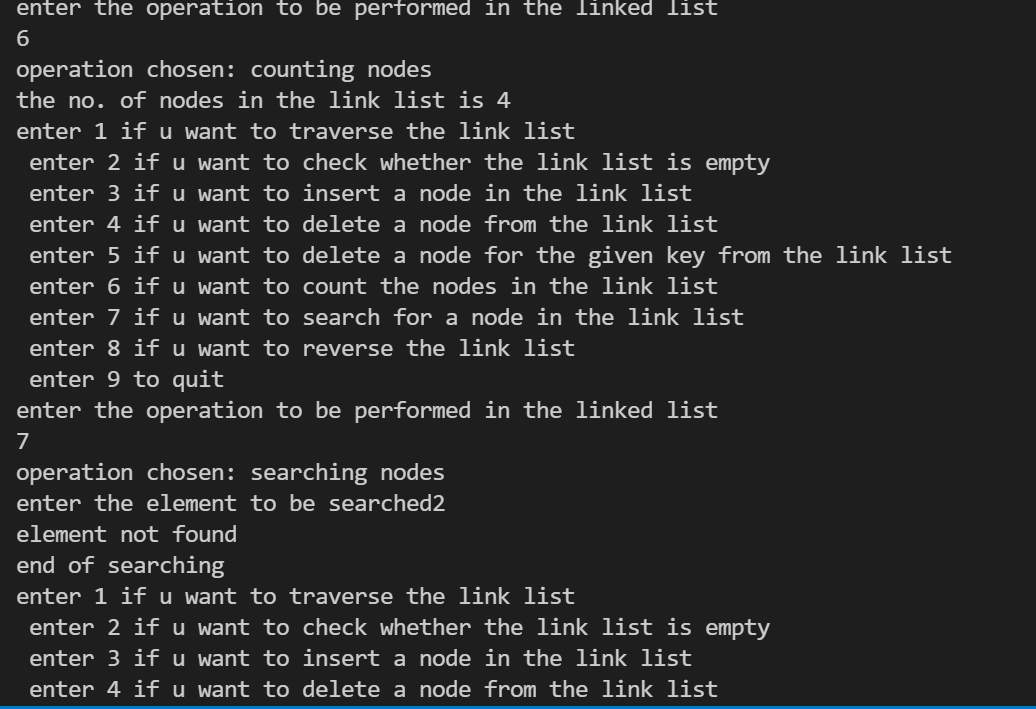


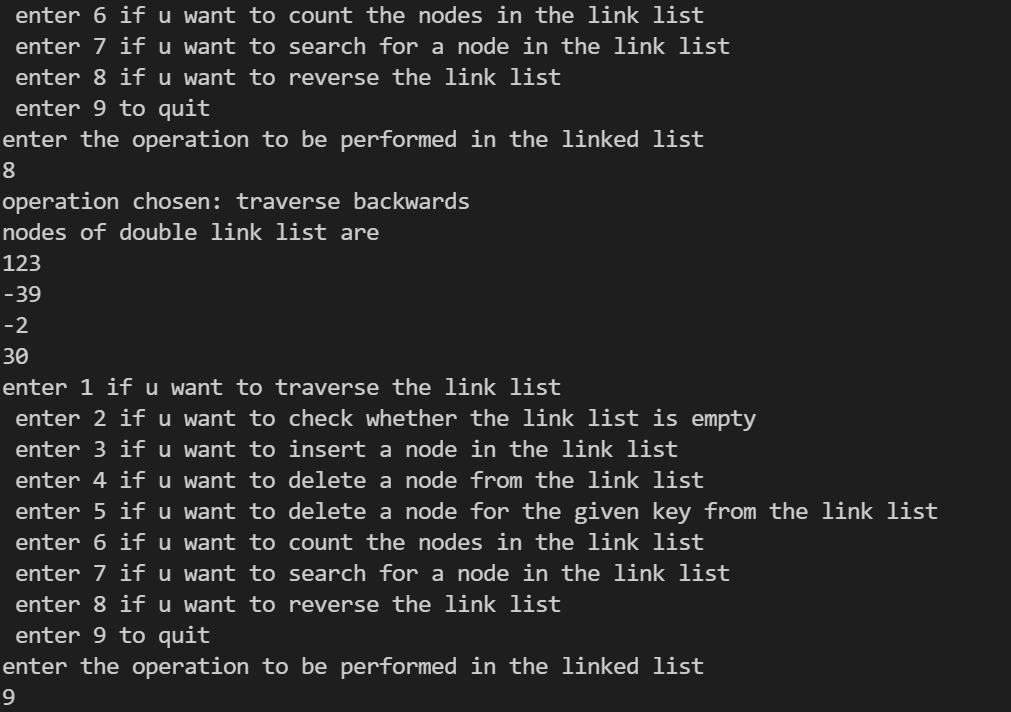












***Question 4: WAP to create a single circular double linked list of n nodes and display the linked list by***

***using suitable user defined functions for create and display operations.***

#include <stdio.h>

#include <stdlib.h>

struct node

{

int value;

struct node \*nextptr;

} \* startnode;

void create(int n);

void display();

int main()

{

int n;

startnode = NULL;

printf(" Input the number of nodes : ");

scanf("%d", &n);

create(n);

display();

return 0;

}

void create(int n)

{

int i, value;

struct node \*preptr, \*newnode;

if (n >= 1)

{

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

printf(" Input data for node 1 : ");

scanf("%d", &value);

startnode->value = value;

startnode->nextptr = NULL;

preptr = startnode;

for (i = 2; i <= n; i++)

{

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

printf(" Input data for node %d : ", i);

scanf("%d", &value);

newnode->value = value;

newnode->nextptr = NULL;

preptr->nextptr = newnode;

preptr = newnode;

}

preptr->nextptr = startnode;

}

}

void display()

{

struct node \*tmp;

int n = 1;

if (startnode == NULL)

{

printf(" No data found in the List yet.\n");

}

else

{

tmp = startnode;

printf(" Displaying the list\n");

do

{

printf(" Data %d = %d\n", n, tmp->value);

tmp = tmp->nextptr;

n++;

} while (tmp != startnode);

}

}

