***NAME - AKRITI CHOUDHARY***

***ROLL NUMBER - 2005776***

***SUBJECT - DSA LAB4***

***DATE - 25/8/2021***

***CLASS - B14***

***BRANCH - CSE***

***Question 1 : WAP to create a linear linked list of n nodes***

***a)display the linked list by using suitable user***

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

***b) display the contents of a linked list in reverse order.***

***c) print mth node from the last of a linked list of n nodes.***

//to create n nodes

//display the data of each node

//display in reverse order using recursion

//to print mth node from the last of a linked list of n nodes.

#include <stdio.h>

#include <stdlib.h>

struct node

{

int data;

struct node \*nextPtr;

} \* startPtr, \*nextNode;

void createList(int n); //to create n nodes

void displayList(int n); //traverse and display the list in forward manner

void revDispRec(struct node \*startPtr); //to display list in reverse order using recursion

void LastDispM(int n, int m); //to print mth node from the last of a linked list of n nodes.

int main()

{

startPtr = NULL;

nextNode = NULL;

int n, m;

printf("Enter the number of nodes to be created : ");

scanf("%d", &n);

createList(n);

puts("Printing in forward manner :");

displayList(n);

puts("Printing in reverse order :");

revDispRec(startPtr);

printf("Enter the index of node from the last to be displayed : ");

scanf("%d", &m);

LastDispM(n, m);

free(startPtr);

free(nextNode);

puts("The memory has been freed successfully");

return 0;

}

void createList(int n)

{

struct node \*temp;

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

if (startPtr == NULL)

{

puts("Memory is not allocated");

}

else

{

puts("Enter data in the node 1");

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

startPtr->nextPtr = NULL;

temp = startPtr;

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

{

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

printf("Enter data in the node : %d \n", i);

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

nextNode->nextPtr = NULL;

temp->nextPtr = nextNode;

temp = nextNode;

}

}

}

void displayList(int n)

{

struct node \*temp = startPtr;

if (startPtr == NULL)

{

puts("The list is empty");

}

else

{

while (temp != NULL)

{

printf("Data : %d \n", temp->data);

printf("address of the next node : %p \n" ,temp->nextPtr);

temp = temp->nextPtr;

}

}

}

void revDispRec(struct node \*startPtr)

{

struct node \*temp = startPtr;

if (temp == NULL)

{

return;

}

else

{

revDispRec(temp->nextPtr);

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

}

}

void LastDispM(int n, int m)

{

struct node \*temp = startPtr;

int count = 0;

if (startPtr == NULL)

{

puts("The list is empty");

}

else

{

while (temp != NULL)

{

count++;

if (count == (n - m + 1))

{

printf("Data : %d \n", temp->data);

break;

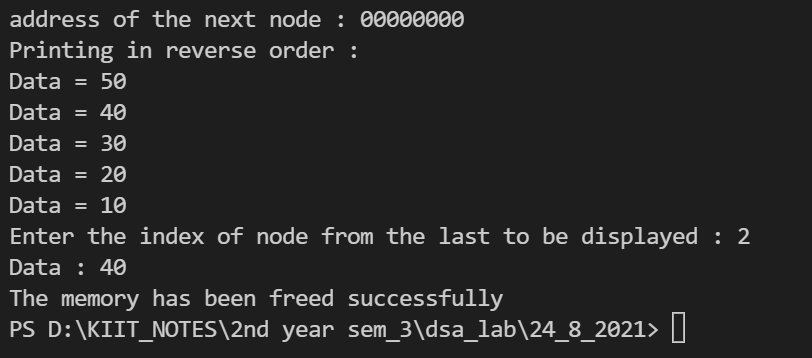
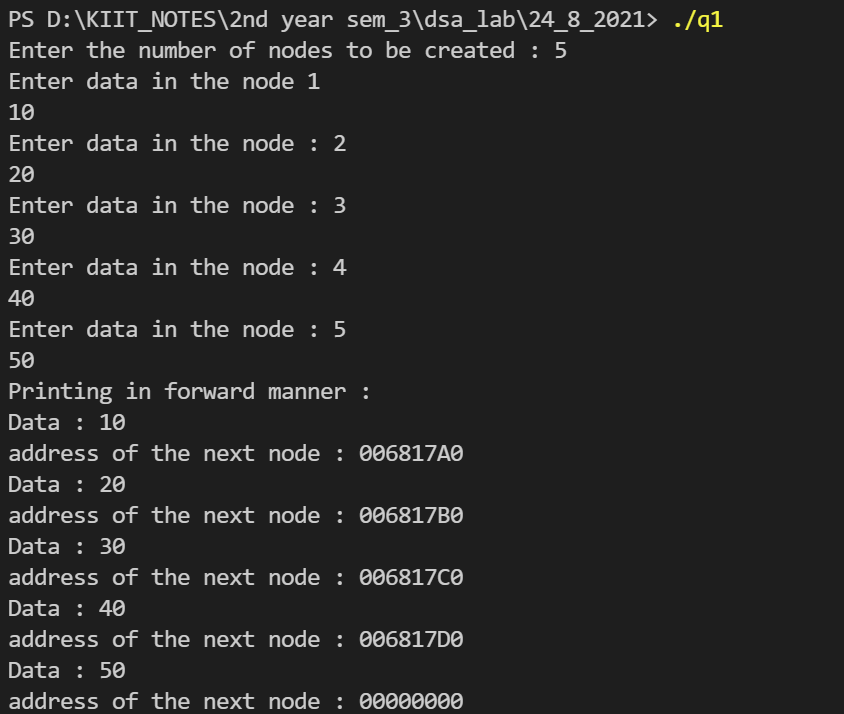
}

temp = temp->nextPtr;

}

}

}



***Question 2 : Write a menu driven program to perform the following operations in a single linked list by***

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

***a) Traversal of the list***

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

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

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

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

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

***g) 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 \* next;

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

void createLinkedList(int n)

{

struct node \* curr;

int x,i=1;

while(i<=n)

{

curr = malloc(sizeof(struct node));

int a;

printf("\n enter the data for node no. %d:",i);

scanf("%d",&a);

curr->data=a;

curr->next = NULL;

if (head == NULL)

{

head = curr;

tail = curr;

}

else

{

tail->next = curr;

tail = curr;

}

i++;

}

}

void displayLinkedList()

{

struct node \* curr;

int c = 0;

curr = head;

while (curr != NULL)

{

c++;

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

curr = curr->next;

}

}

void checkEmpty()

{

struct node \* ptr;

ptr=head;

if (ptr== NULL)

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

else

printf("linked list is not empty\n");

}

void insertNode(int n)

{

struct node \* ptr;

int b;

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

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

scanf("%d",&b);

if (n == 1)

{

curr->data = b;

curr->next = head;

head = curr;

}

else

{ curr->data = b;

ptr = head;

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

ptr=ptr->next;

curr->next=ptr->next;

ptr->next=curr;

}

printf("node inserted\n");

displayLinkedList();

}

void deleteNode(int pos,int size)

{

struct node \* ptr;

int j=1;

ptr =head;

if(pos>1 && pos<=size )

{

while (ptr!=NULL)

{

if(j==pos-1)

{

ptr->next=((ptr->next)->next);

break;

}

else

{

ptr=ptr->next;

j++;

}

}

printf("end of deletion from position %d \n",pos);

displayLinkedList();

}

if(pos==1)

{

head=head->next;

printf("end of deletion from beginning\n");

displayLinkedList();

}

if(pos>size)

{

printf("nothing to delete\n");

}

}

int countNode()

{struct node \* ptr;

ptr=head;

int x=0;

while(ptr!=NULL)

{

x++;

ptr=ptr->next;

}

return x;

}

void SearchNode(int 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 reverseNodes()

{

struct node \* prev=NULL,\* curr,\* nt=NULL;

curr=head;

while (curr!=NULL)

{

nt=curr->next;

curr->next=prev;

prev=curr;

curr=nt;

}

head=prev;

displayLinkedList();

}

void deletekey()

{

struct node \* temp=head;

struct node \* p=NULL;

int key;

printf("enter the key");

scanf("%d",&key);

while(temp!=NULL)

{

if(temp->data==key)

{

if(temp==head)

{

head=head->next;

temp=head;

}

else

{

p->next=temp->next;

temp=p->next;

}

}

else

{

p=temp;

temp=temp->next;

}

}

if(temp==NULL)

displayLinkedList();

}

int main()

{

int n, op, no,pos,src;

printf("Enter Number of elements you want to enter in the linked list\n");

scanf("%d", &no);

createLinkedList(no);

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 count the nodes in the link list\n ");

printf("enter 6 if u want to search for a node in the link list\n ");

printf("enter 7 if u want to reverse the link list\n ");

printf("enter 8 if u want to delete a node for the given key from 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");

displayLinkedList();

break;

case 2:

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

checkEmpty();

break;

case 3:

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

printf("Enter The position where you want to insert the node\n");

scanf("%d", &n);

insertNode(n);

break;

case 4:

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

printf("enter position to delete \n");

scanf("%d",&pos);

deleteNode(pos,no);

break;

case 5:

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

printf("Number of nodes = %d\n",countNode());

break;

case 6:

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

printf("Enter Search element = ");

scanf("%d",&src);

SearchNode(src);

break;

case 7:

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

reverseNodes();

break;

case 8:

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

deletekey();

break;

case 9:

break;

}

}while(op!=9);

}

