**TITLE 43**

Write a C program to perform the following tasks:

a. Insert a node at the beginning of a singly-linked list.

b. Insert a node at end of a singly-linked list.

c. Insert a node at the middle of a singly-linked list.

d. Delete a node from the beginning of the singly-linked list.

e. Delete a node from the end of a singly-linked list

**OBJECTIVE:**

By the end of this problem we will be able to insert a node at the beginning, middle and the end of a singly-linked list and also delete a node from the beginning and the end of a singly-linked list.

**PROBLEM STATEMENT:**

In this program we will perform certain operations on the singly linked list like inserting a node at the beginning, middle and the end and deleting a node from the beginning and the end of a singly linked list.

Once the input is collected and stored the output is printed.

**ALGORITHM:**

START

Define variables: data, \*next, \*prev

INPUT: Read from the keyboard

COMPUTATION: Computing different operations on the singly-linked list

DISPLAY: Displaying the output after the different operations are performed on a singly-linked list.

STOP

**PROGRAM:**

#include <stdio.h>

#include <stdlib.h>

#define init() ((struct node\*)malloc(sizeof(struct node)))

typedef struct node

{

int data;

struct node \*next;

struct node \*prev;

}node;

//function which creates nodes

node\* createNode(int data){

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

n->data = data;

n->next = NULL;

return n;

}

//inserts a node in the begging

node\* insertBeg(node \*head,int data){

node \*newNode = createNode(data);

newNode->next = head;

return newNode;

}

//inserts at middle

//if nodes = even it adds at nodes/2

//if nodes = odd adds at nodes/2 + 1

void insertMiddle(node \*head,int data){

node \*ptr = head;

if(ptr == NULL){

//if list empty doesnot add element

printf("empty\n");

return;

}

//runns till the end

while(head->next != NULL){

if(head->next->next != NULL){

head = head->next->next;//runs fast

ptr = ptr->next;//runs half the iterations

}

else{

break;

}

}

node \*temp = ptr->next;

ptr->next = createNode(data);

ptr->next->next = temp;

}

//recursive function which insert the node at the end

node\* insertEnd(node \*head, int data){

//if empty return a newlist with one element

if(head == NULL){

return createNode(data);

}

//calls insertend function

head->next = insertEnd(head->next, data);

return head;

}

//deletes node in the beggining

node\* deleteBeg(node \*head){

node \*temp = head;

//node empty thus returns null

if(head == NULL){

printf("Empty\n");

return NULL;

}

//deletes the node

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

free(temp);

//returns the head's next elements

return head->next;

}

//deletes the last last node(not recursive)

node\* deleteEnd(node \*head){

//if empty list return NULL

if(head == NULL ){

printf("empty\n");

return NULL;

}

//if a single element returns NULL

if (head->next == NULL){

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

free(head);

return NULL;

}

//last node deleted

head->next = deleteEnd(head->next);

return head;

}

//displays the list

void display(node \*head){

while(head!=NULL){

printf("%d->",head->data);

head = head->next;

}

printf("NULL\n");

}

//main

int main () {

int choice,data;

node \*head ;

while(1){

//menu

printf("\n\*\*\*Main Menu\*\n");

printf("\nChoose one option from the following list ...\n");

printf("\n===============================================\n");

printf("\n1.Insert in beginning\n2.Insert at last\n3.Insert middle.\n4.Delete number at the beginning \n5.Delete number at the end\n6.Display\n7.Exit\n");

printf("\nEnter your choice?\n");

scanf("\n%d",&choice);

//performs operation according to the choice

switch(choice){

case 1:{

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

scanf("%d",&data);

head = insertBeg(head, data);

break;

}

case 2:{

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

scanf("%d",&data);

head = insertEnd(head, data);

break;

}

case 3:{

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

scanf("%d",&data);

insertMiddle(head, data);

break;

}

case 4:{

head = deleteBeg(head);

break;

}

case 5:{

head = deleteEnd(head);

break;

}

case 6:{

printf("The list:\n");

display(head);

break;

}

case 7: {exit(0);break;}

}

}

return 0;

}

**CONCLUSION:**

The simulation of the above program helps us change a singly-linked list according to the users choice.

**OUTPUT:**

\*\*\*Main Menu\*

Choose one option from the following list ...

===============================================

1.Insert in beginning

2.Insert at last

3.Insert middle

4.Delete number at the beginning

5.Delete number at the end

6.Display

7.Exit

Enter your choice?

1

Enter the data to be inserted

6

\*\*\*Main Menu\*

Choose one option from the following list ...

===============================================

1.Insert in beginning

2.Insert at last

3.Insert middle

4.Delete number at the beginning

5.Delete number at the end

6.Display

7.Exit

Enter your choice?

1

Enter the data to be inserted

7

\*\*\*Main Menu\*

Choose one option from the following list ...

===============================================

1.Insert in beginning

2.Insert at last

3.Insert middle

4.Delete number at the beginning

5.Delete number at the end

6.Display

7.Exit

Enter your choice?

1

Enter the data to be inserted

8

\*\*\*Main Menu\*

Choose one option from the following list ...

===============================================

1.Insert in beginning

2.Insert at last

3.Insert middle

4.Delete number at the beginning

5.Delete number at the end

6.Display

7.Exit

Enter your choice?

6

The list:

8->7->6->NULL

\*\*\*Main Menu\*

Choose one option from the following list ...

===============================================

1.Insert in beginning

2.Insert at last

3.Insert middle

4.Delete number at the beginning

5.Delete number at the end

6.Display

7.Exit

Enter your choice?

7