**#5. Linked List**

**Roll Number: CB.EN.P2EBS22001**

**Date of Submission: 25-11-2022**

**Aim:**

To perform following operations in a Linked List using C Programming:

1. Creation of linked list
2. Addition of nodes in the list (start, end and middle)
3. Deletion of nodes in the list (start, end and middle)
4. Print the entire list after each of the above operation

**Tools Required:**

Text editor with C Compiler.

**Experiment:**

Code

// Singly Linked List

#include <stdio.h>

#include <stdlib.h>

typedef struct node {

    int info;

    struct node\* link;

}node;

struct node\* start = NULL;

void createList()

{

    if (start == NULL) {

        int n;

        printf("\nEnter the number of nodes: ");

        scanf("%d", &n);

        if (n != 0) {

            int data;

            struct node\* newnode;

            struct node\* temp;

            newnode = malloc(sizeof(node));

            start = newnode;

            temp = start;

            printf("\nEnter number to be inserted : ");

            scanf("%d", &data);

            start->info = data;

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

                newnode = malloc(sizeof(node));

                temp->link = newnode;

                printf("\nEnter number to be inserted : ");

                scanf("%d", &data);

                newnode->info = data;

                temp = temp->link;

            }

        }

        printf("\nThe list is created\n");

    }

    else

        printf("\nThe list is already created\n");

}

// Function to traverse the linked list

void traverse()

{

    struct node\* temp;

    // List is empty

    if (start == NULL)

        printf("\nList is empty\n\n");

    else {

        temp = start;

        while (temp != NULL) {

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

            temp = temp->link;

        }

    }

}

// Function to insert at the front of the linked list

void insertAtFront()

{

    int data;

    struct node\* temp;

    temp = malloc(sizeof(node));

    printf("\nEnter number to"

        " be inserted : ");

    scanf("%d", &data);

    temp->info = data;

    temp->link = start;

    start = temp;

}

// Function to insert at the end of the linked list

void insertAtEnd()

{

    int data;

    struct node \*temp, \*head;

    temp = malloc(sizeof(struct node));

    // Enter the number

    printf("\nEnter number to"

        " be inserted : ");

    scanf("%d", &data);

    // Changes links

    temp->link = 0;

    temp->info = data;

    head = start;

    while (head->link != NULL) {

        head = head->link;

    }

    head->link = temp;

}

// Function to insert at any specified position

void insertAtPosition()

{

    struct node \*temp, \*newnode;

    int pos, data, i = 1;

    newnode = malloc(sizeof(struct node));

    // Enter the position and data

    printf("\nEnter position and data:");

    printf("\nEnter the data :");

    scanf("%d %d", &pos, &data);

    // Change Links

    temp = start;

    newnode->info = data;

    newnode->link = 0;

    while (i < pos - 1) {

        temp = temp->link;

        i++;

    }

    newnode->link = temp->link;

    temp->link = newnode;

}

// Function to delete from the front of the linked list

void deleteFirst()

{

    struct node\* temp;

    if (start == NULL)

        printf("\nList is empty\n");

    else {

        temp = start;

        start = start->link;

        free(temp);

    }

}

// Function to delete from the end of the linked list

void deleteEnd()

{

    struct node \*temp, \*prevnode;

    if (start == NULL)

        printf("\nList is Empty\n");

    else {

        temp = start;

        while (temp->link != 0) {

            prevnode = temp;

            temp = temp->link;

        }

        free(temp);

        prevnode->link = 0;

    }

}

// Function to delete from any specified position from the linked list

void deletePosition()

{

    struct node \*temp, \*position;

    int i = 1, pos;

    if (start == NULL)

        printf("\nList is empty\n");

    else {

        printf("\nEnter index : ");

        // Position to be deleted

        scanf("%d", &pos);

        position = malloc(sizeof(struct node));

        temp = start;

        while (i < pos - 1) {

            temp = temp->link;

            i++;

        }

        position = temp->link;

        temp->link = position->link;

        free(position);

    }

}

int main()

{

    int choice;

    while (1) {

        printf("\t\t=====OPTIONS=====");

        printf("\n\t1) View the list\n");

        printf("\t2) Insert at the beginning \n");

        printf("\t3) Insert at the end\n");

        printf("\t4) Insertion at any position\n");

        printf("\t5) Delete the first element\n");

        printf("\t6) Delete the last element\n");

        printf("\t7) Delete element at any position\n");

        printf("\n8) To exit\n");

        printf("\nEnter Choice :");

        scanf("%d", &choice);

        switch (choice) {

        case 1:

            traverse();

            break;

        case 2:

            insertAtFront();

            break;

        case 3:

            insertAtEnd();

            break;

        case 4:

            insertAtPosition();

            break;

        case 5:

            deleteFirst();

            break;

        case 6:

            deleteEnd();

            break;

        case 7:

            deletePosition();

            break;

        case 8:

            exit(1);

            break;

        default:

            printf("Incorrect Choice\n");

        }

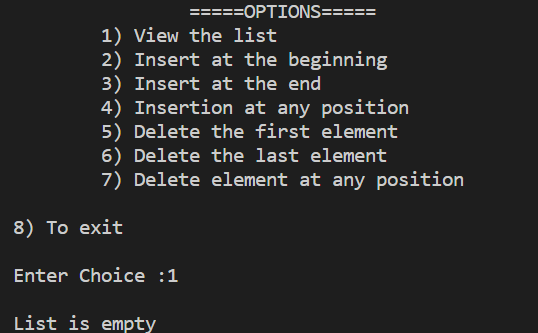
    }

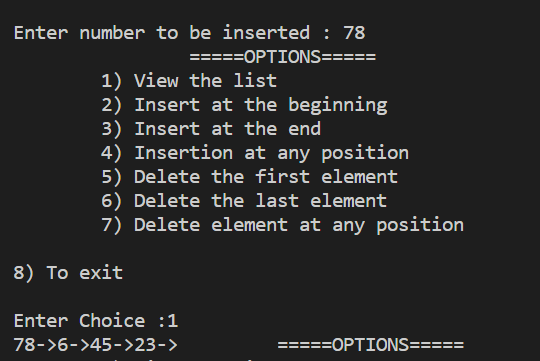
    return 0;

}

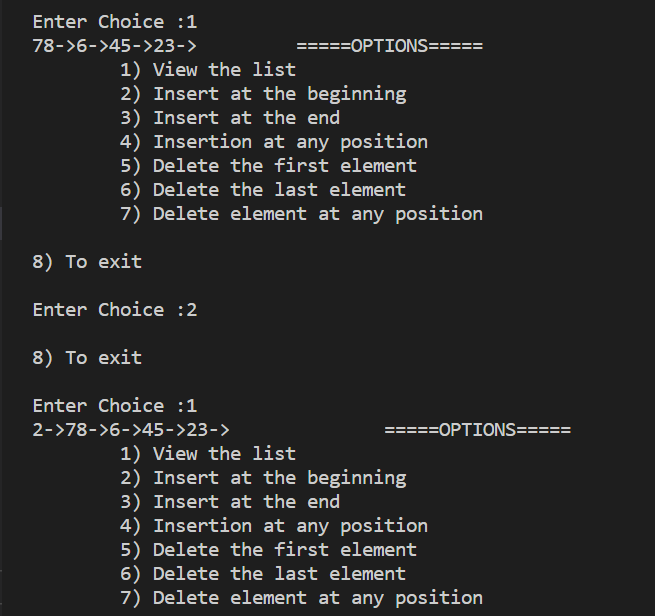
Result

1. Creation of Linked List (78->6->45->23)

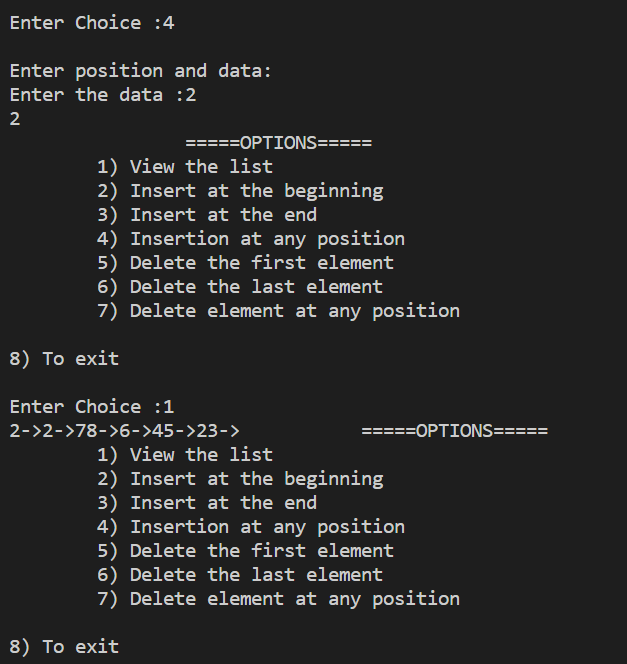




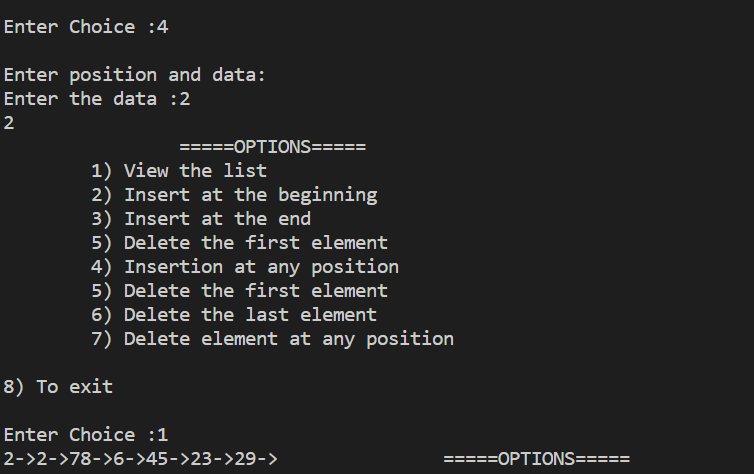
1. Addition of nodes in the list (start, end and middle)
2. Start



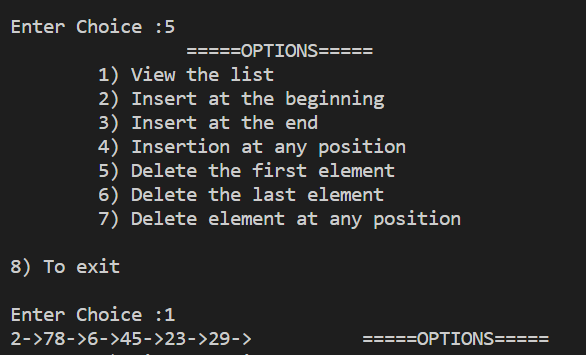
1. Middle



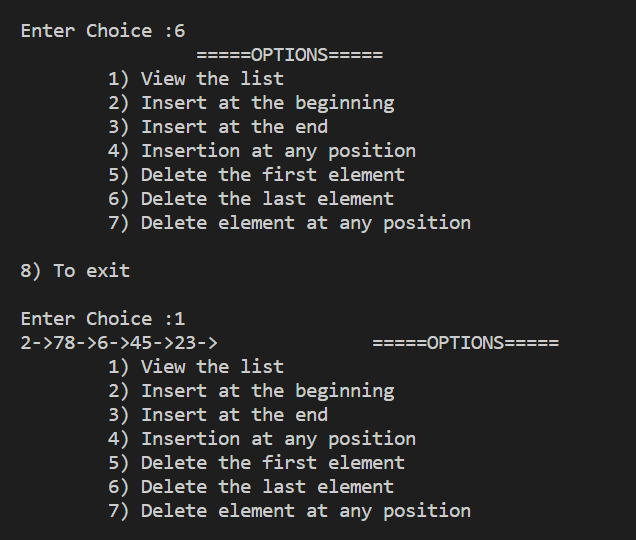
1. End



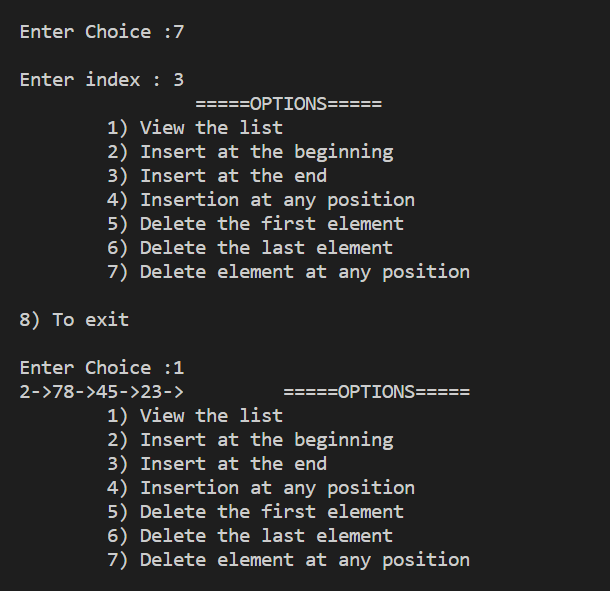
1. Deletion of nodes in the list (start, end and middle)
2. Start



1. End



1. Middle



**Inference and Result**

Singly Linked list is implemented using C programming and linked list operations are performed.