```
#include <stdio.h>
#include <stdlib.h>

void main(){

    typedef struct LinkedList
    {
       int data;
       struct LinkedList *next;
    }Node;

    int choice;
    Node *new,*head,*ptr,*temp,*ptr2,*prev;
    int DATA,pos;
    head = NULL;

    while (1){
    printf("\n\n");
    printf("SELECT AN OPERATION:\n");
    printf("1.Display\n");
    printf("2.Insert at beginning\n");
    printf("3.Insert at end\n");
    printf("4.Insert at specified position\n");
    printf("5.Delete from beginning\n");
    printf("6.Delete from end\n");
    printf("7.Delete from a specified position\n");
    scanf("%d",&choice);

    switch(choice){
        // Head is a pointer to the fist element of the LL
        // NOT a separate node
        ptr = head;

        case 1:
            if (head == NULL){
                printf("Empty list");
            }
            else{
                printf("The elements are: ");
                while(ptr != NULL){ // To check till the last element
                  printf("%d ",ptr->data);
                  ptr = ptr->next;
            }}
            break;

        case 2:
            printf("Enter the data to be inserted:");
            scanf("%d",DATA);
            new = (Node*) malloc(sizeof(int));
```

```
    if (head == NULL){
        head = new;
        head->data = DATA;
        head->next = NULL;
        printf("Insertion successful");

    }

    else{
        temp = head->next;
        head = new;
        head->next = temp;
        head->data = DATA;
        printf("Insertion successful");

    }
    break;

case 3:
    printf("Enter the data to be inserted:");
    scanf("%d",DATA);
    new = (Node*) malloc(sizeof(int));
    if (head == NULL){
        head = new;
        head->data = DATA;
        head->next = NULL;
        printf("Insertion successful");
    }

    else{
       while(ptr->next != NULL){
            ptr = ptr->next;
       }
       ptr->next = new;
       new->data = DATA;
       new->next = NULL;
       printf("Insertion successful");
    }
    break;

case 4:
    printf("Enter the data to be inserted:");
    scanf("%d",DATA);
    new = (Node*) malloc(sizeof(int));
    printf("Enter the position at which the data is to be added:");
    scanf("%d",&pos);
    int len=1;
    ptr2 = ptr;
    while(ptr2->next!=NULL){
        ptr2 = ptr2->next;
```

```
        len++;
    }
    if (pos >= len)
        printf("Out of range");
    else{
    int ct = 1;
    while(ct<pos){
        ptr = ptr->next;
        ct++;
    }
    temp = ptr->next;
    ptr->next = new;
    new->data = DATA;
    new->next = temp;
    }
    break;

case 5:

    if (head == NULL){
        printf("No elements to delete");
    }
    else{
        head = head->next;
        printf("%d deleted",ptr->data);
        free(ptr);
    }
    break;

case 6:
    if (head == NULL){
        printf("No elements to delete");
    }
    else{
        while(ptr->next != NULL){
            prev = ptr;
            ptr = ptr->next;
        }
        prev->next = NULL;
        printf("%d deleted",ptr->data);
        free(ptr);
    }
    break;
case 7:
    if (head == NULL){
        printf("No elements to delete");
    }
    else{
    printf("Enter the position at which the data is to be added:");
    scanf("%d",&pos);
```

```
            int len=1;
            ptr2 = ptr;
            while(ptr2->next!=NULL){
                ptr2 = ptr2->next;
                len++;
            }
            if (pos >= len)
                printf("Out of range");
            else{
                int ct =1;
                while(ct<pos){
                    temp = ptr;
                    ptr = ptr->next;
                    printf("%d deleted",temp->data);
                    free(temp);
                    ct++;
                }
            }
            }
            break;
        default:
            printf("Wrong input.");
            break;
    }
    }
}
```