Laboratory 7

1. **Questions**

a) Implement a linked list and illustrate the following operations.
I. Insert a node at the beginning
II. Insert a node at the end
III. Print the linked list
b) Write a program to create a linked list and delete the element entered by a user.

2. **Algorithm**

1.**Algorithm to Implement a linked list operations of insertion of element at end of node and at beginning**

Inserting At Beginning of the list
Step 1: Create a newNode with given value.
Step 2: Check whether list is Empty (head == NULL)
Step 3: If it is Empty then, set newNode→next = NULL and head = newNode.
Step 4: If it is Not Empty then, set newNode→next = head and head = newNode.

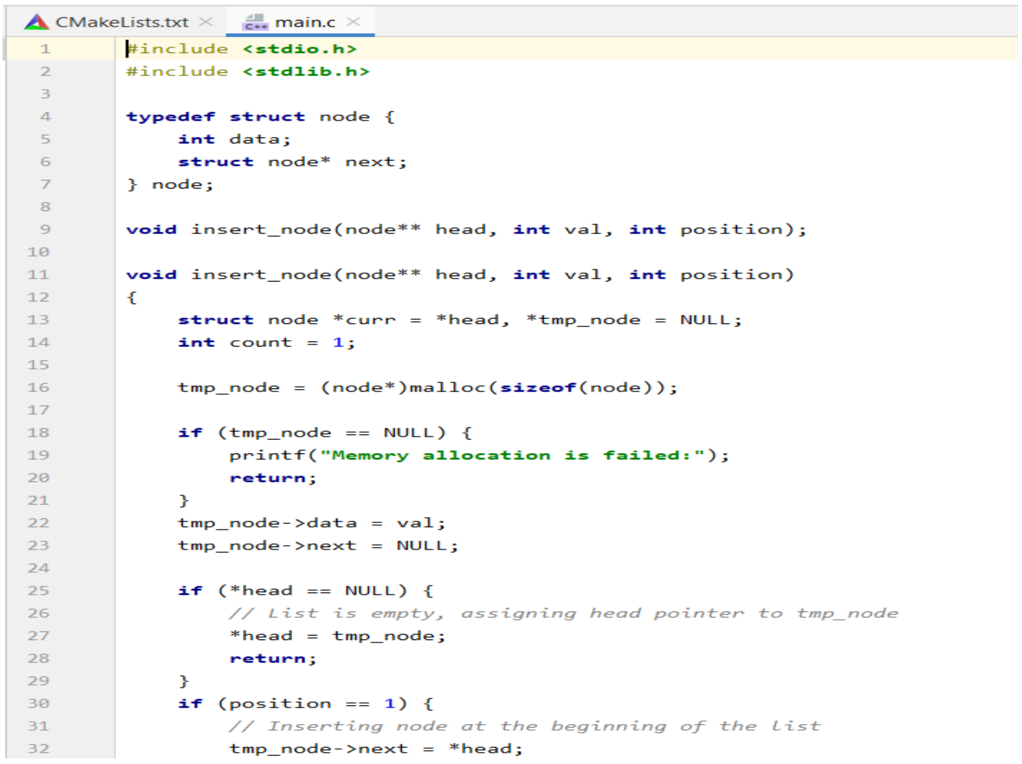
Inserting At End of the list
Step 1: Create a newNode with given value and newNode → next as NULL.
Step 2: Check whether list is Empty (head == NULL).
Step 3: If it is Empty then, set head = newNode.
Step 4: If it is Not Empty then, define a node pointer temp and initialize with head.
Step 5: Keep moving the temp to its next node until it reaches to the last node in the list (until temp → next is equal to NULL).
Step 6: Set temp → next = newNode.

Inserting At Specific location in the list (After a Node)
Step 1: Create a newNode with given value.
Step 2: Check whether list is Empty (head == NULL)
Step 3: If it is Empty then, set newNode → next = NULL and head = newNode.
Step 4: If it is Not Empty then, define a node pointer temp and initialize with head.
Step 5: Keep moving the temp to its next node until it reaches to the node after which we want to insert the newNode (until temp1 → data is equal to location, here location is the node value after which we want to insert the newNode).
Step 6: Every time check whether temp is reached to last node or not. If it is reached to last node then display 'Given node is not found in the list!!! Insertion not possible!!!' and terminate the function. Otherwise move the temp to next node.
Step 7: Finally, Set 'newNode → next = temp → next' and 'temp → next = newNode'

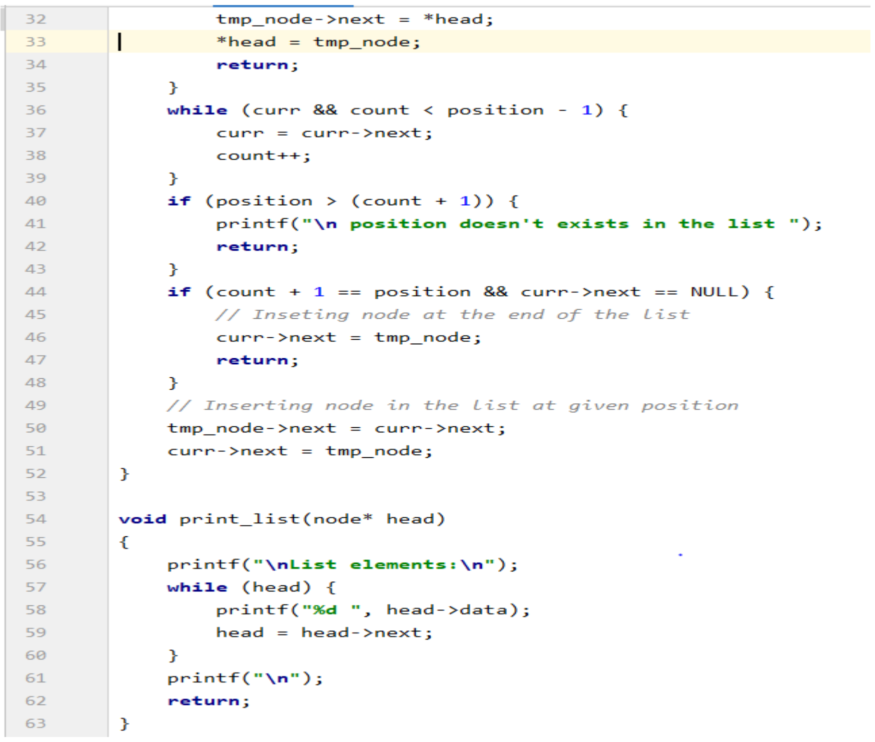
2.Algorithm for program to create a linked list and delete the element entered by a user.
Deleting a Specific Node from the list
Step 1: Check whether list is Empty (head == NULL)
Step 2: If it is Empty then, display 'List is Empty!!! Deletion is not possible' and terminate the function.
Step 3: If it is Not Empty then, define two Node pointers 'temp1' and 'temp2' and initialize 'temp1' with head.
Step 4: Keep moving the temp1 until it reaches to the exact node to be deleted or to the last node. And every time set 'temp2 = temp1' before moving the 'temp1' to its next node.
Step 5: If it is reached to the last node then display 'Given node not found in the list! Deletion not possible!!!'. And terminate the function.
Step 6: If it is reached to the exact node which we want to delete, then check whether list is having only one node or not
Step 7: If list has only one node and that is the node to be deleted, then set head = NULL and delete temp1 (free(temp1)).
Step 8: If list contains multiple nodes, then check whether temp1 is the first node in the list (temp1 == head).

Step 9: If temp1 is the first node then move the head to the next node (head = head → next) and delete temp1.
Step 10: If temp1 is not first node then check whether it is last node in the list (temp1 → next == NULL).
Step 11: If temp1 is last node then set temp2 → next = NULL and delete temp1 (free(temp1)).
Step 12: If temp1 is not first node and not last node then set temp2 → next = temp1 → next and delete temp1 (free(temp1)).

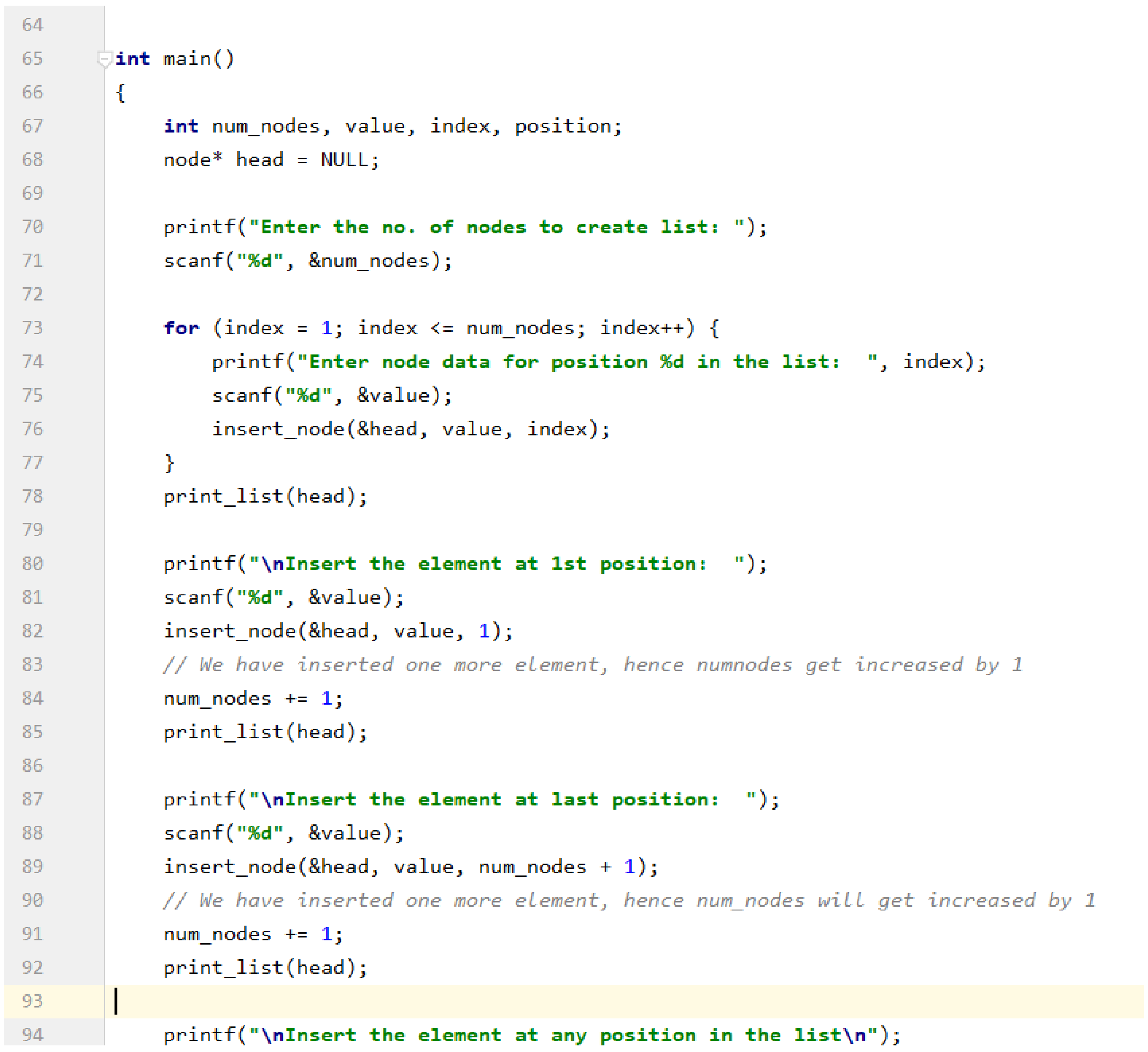
3. Program



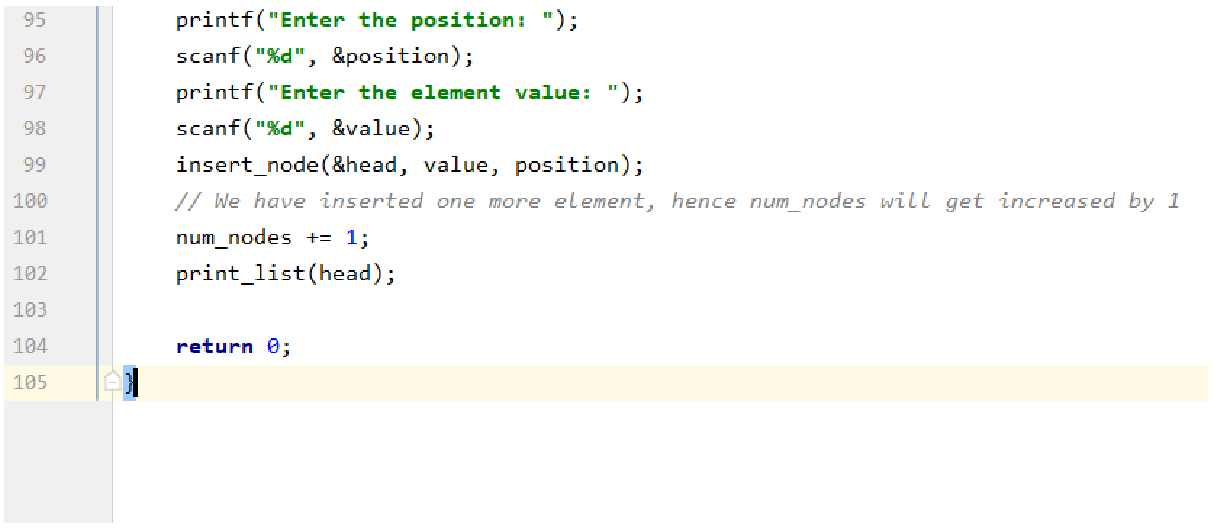
```c
#include <stdio.h>
#include <stdlib.h>

typedef struct node {
    int data;
    struct node* next;
} node;

void insert_node(node** head, int val, int position);

void insert_node(node** head, int val, int position)
{
    struct node *curr = *head, *tmp_node = NULL;
    int count = 1;

    tmp_node = (node*)malloc(sizeof(node));

    if (tmp_node == NULL) {
        printf("Memory allocation is failed:");
        return;
    }
    tmp_node->data = val;
    tmp_node->next = NULL;

    if (*head == NULL) {
        // List is empty, assigning head pointer to tmp_node
        *head = tmp_node;
        return;
    }
    if (position == 1) {
        // Inserting node at the beginning of the list
        tmp_node->next = *head;
```



```c
        tmp_node->next = *head;
        *head = tmp_node;
        return;
    }
    while (curr && count < position - 1) {
        curr = curr->next;
        count++;
    }
    if (position > (count + 1)) {
        printf("\n position doesn't exists in the list ");
        return;
    }
    if (count + 1 == position && curr->next == NULL) {
        // Inseting node at the end of the list
        curr->next = tmp_node;
        return;
    }
    // Inserting node in the list at given position
    tmp_node->next = curr->next;
    curr->next = tmp_node;
}

void print_list(node* head)
{
    printf("\nList elements:\n");
    while (head) {
        printf("%d ", head->data);
        head = head->next;
    }
    printf("\n");
    return;
}
```



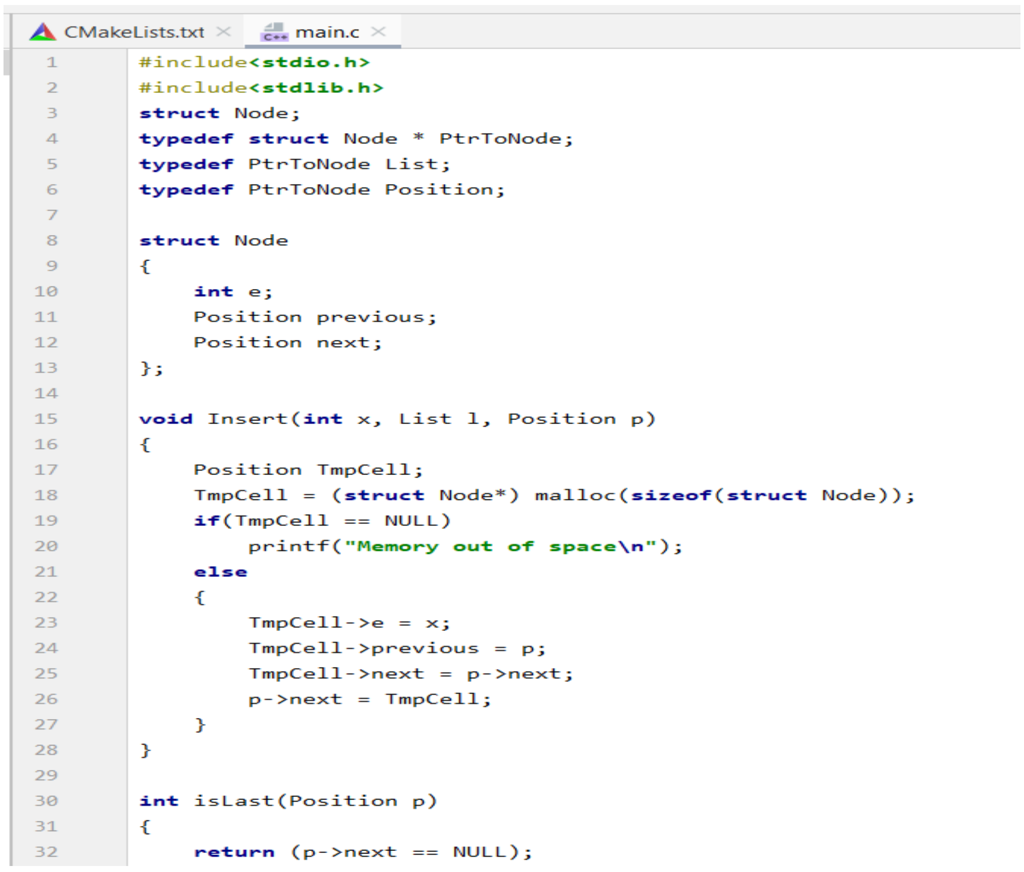
```

int main()
{
    int num_nodes, value, index, position;
    node* head = NULL;

    printf("Enter the no. of nodes to create list: ");
    scanf("%d", &num_nodes);

    for (index = 1; index <= num_nodes; index++) {
        printf("Enter node data for position %d in the list:  ", index);
        scanf("%d", &value);
        insert_node(&head, value, index);
    }
    print_list(head);

    printf("\nInsert the element at 1st position:  ");
    scanf("%d", &value);
    insert_node(&head, value, 1);
    // We have inserted one more element, hence numnodes get increased by 1
    num_nodes += 1;
    print_list(head);

    printf("\nInsert the element at last position:  ");
    scanf("%d", &value);
    insert_node(&head, value, num_nodes + 1);
    // We have inserted one more element, hence num_nodes will get increased by 1
    num_nodes += 1;
    print_list(head);

    printf("\nInsert the element at any position in the list\n");
```



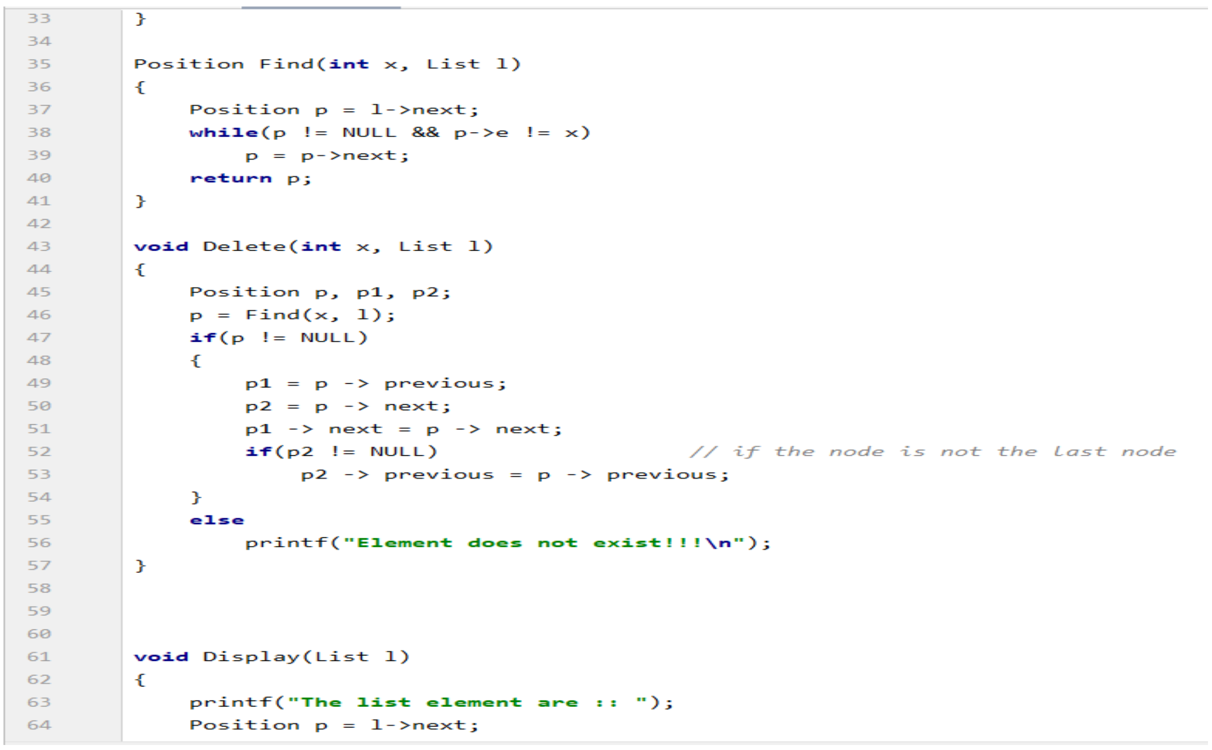
```
    printf("Enter the position: ");
    scanf("%d", &position);
    printf("Enter the element value: ");
    scanf("%d", &value);
    insert_node(&head, value, position);
    // We have inserted one more element, hence num_nodes will get increased by 1
    num_nodes += 1;
    print_list(head);

    return 0;
}
```

Fig 1 program to Implement a linked list operations of insertion of element at end of node and at beginning

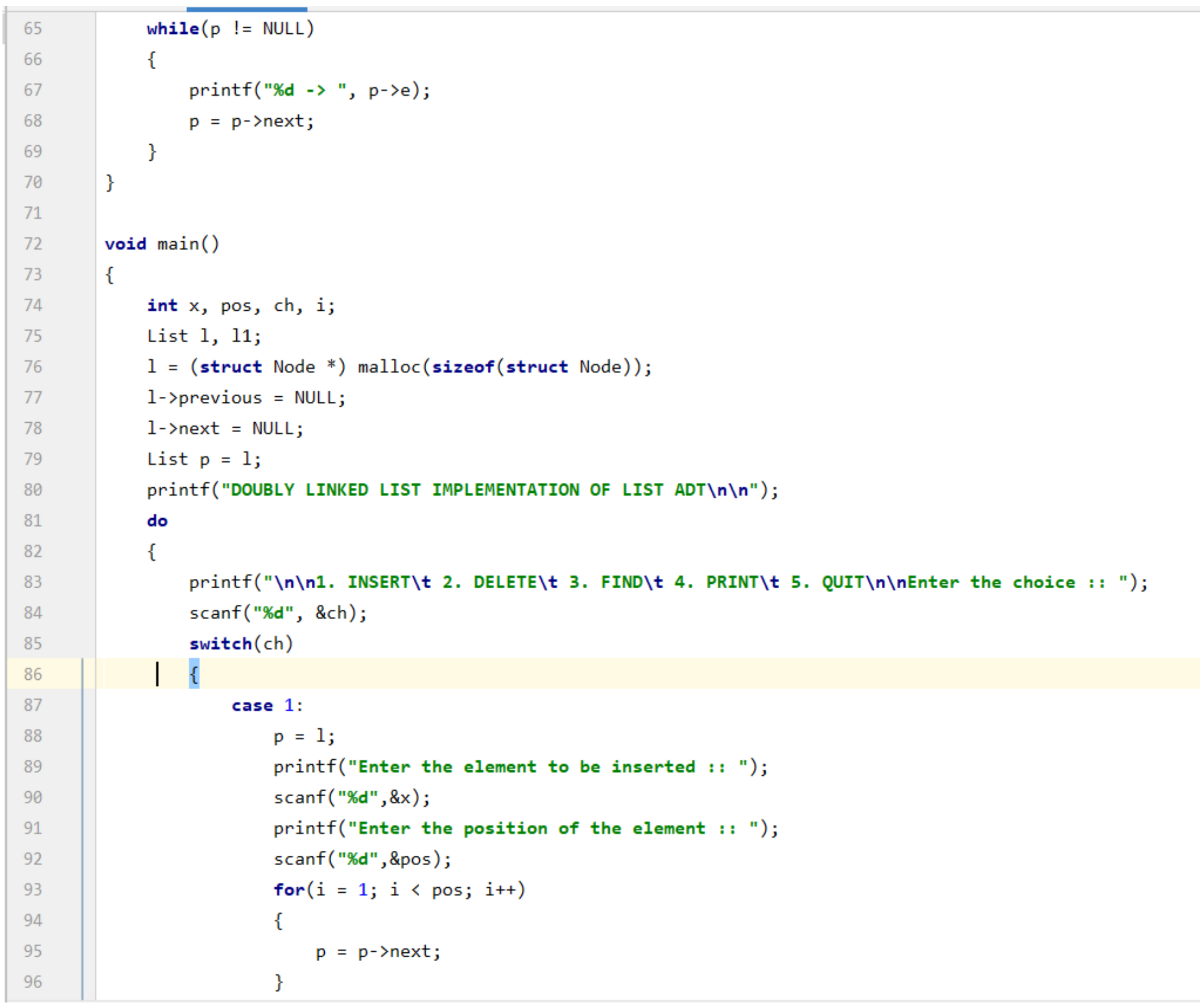
DEEPAK R 18ETCS002041



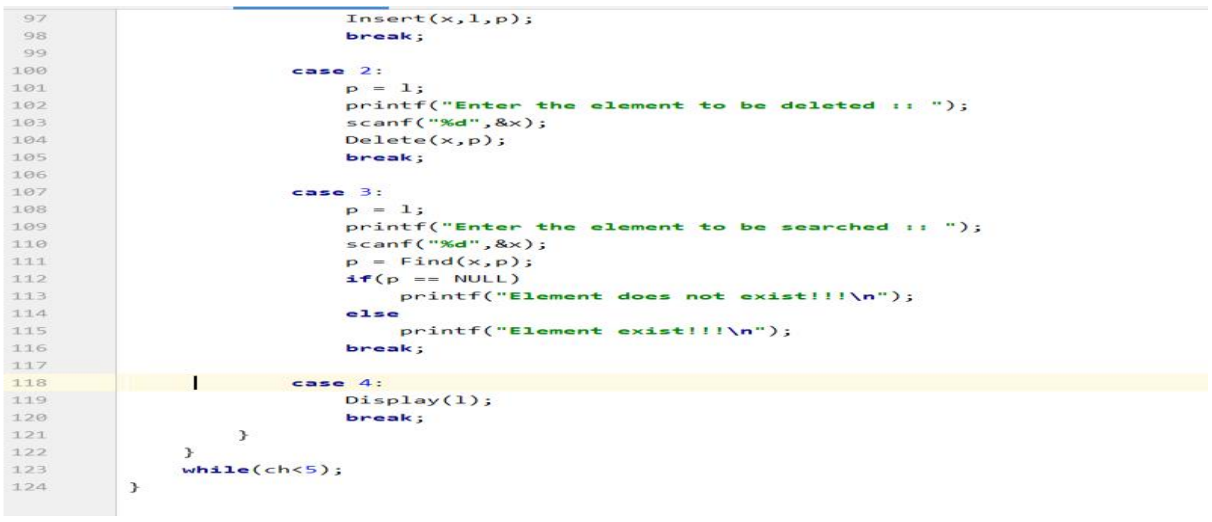
```c
#include<stdio.h>
#include<stdlib.h>
struct Node;
typedef struct Node * PtrToNode;
typedef PtrToNode List;
typedef PtrToNode Position;

struct Node
{
    int e;
    Position previous;
    Position next;
};

void Insert(int x, List l, Position p)
{
    Position TmpCell;
    TmpCell = (struct Node*) malloc(sizeof(struct Node));
    if(TmpCell == NULL)
        printf("Memory out of space\n");
    else
    {
        TmpCell->e = x;
        TmpCell->previous = p;
        TmpCell->next = p->next;
        p->next = TmpCell;
    }
}

int isLast(Position p)
{
    return (p->next == NULL);
```



```c
}

Position Find(int x, List l)
{
    Position p = l->next;
    while(p != NULL && p->e != x)
        p = p->next;
    return p;
}

void Delete(int x, List l)
{
    Position p, p1, p2;
    p = Find(x, l);
    if(p != NULL)
    {
        p1 = p -> previous;
        p2 = p -> next;
        p1 -> next = p -> next;
        if(p2 != NULL)                      // if the node is not the last node
            p2 -> previous = p -> previous;
    }
    else
        printf("Element does not exist!!!\n");
}



void Display(List l)
{
    printf("The list element are :: ");
    Position p = l->next;
```



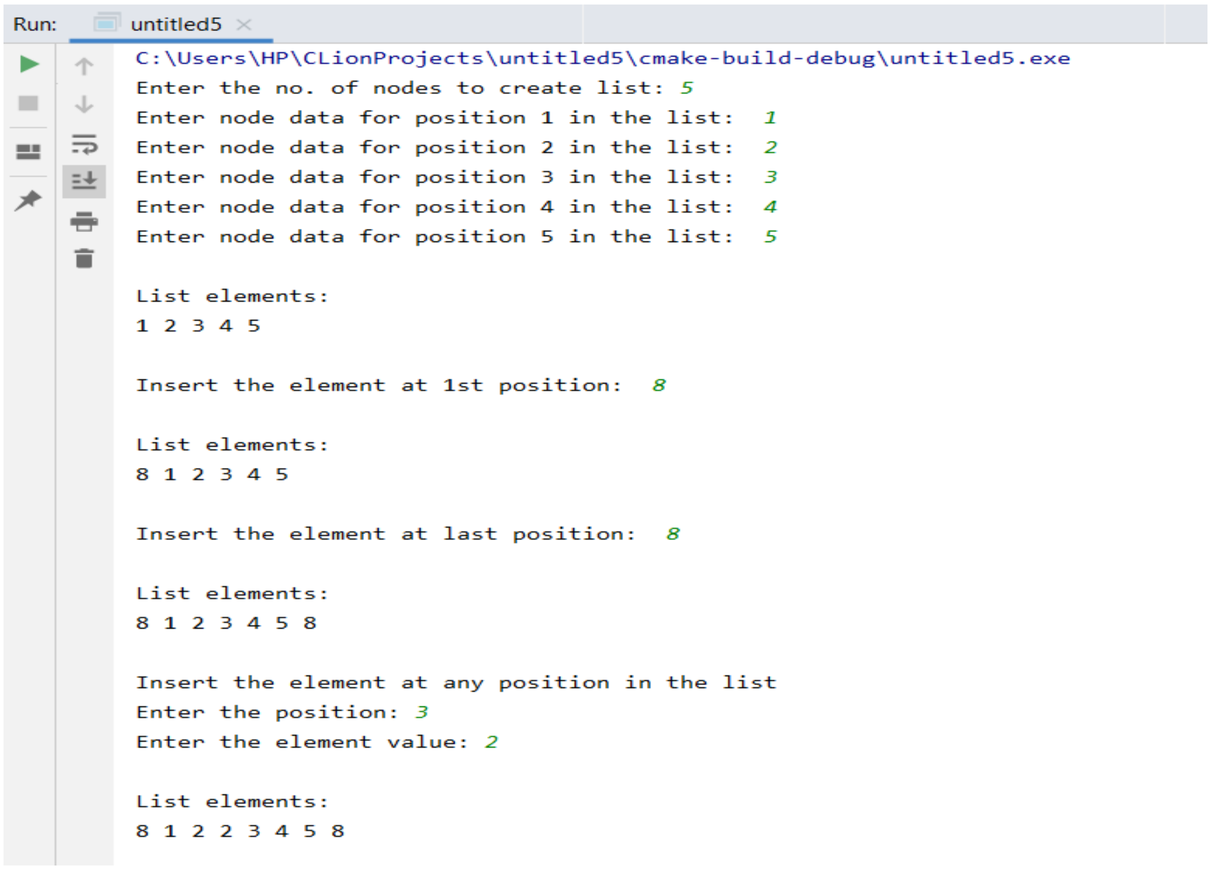
```
    while(p != NULL)
    {
        printf("%d -> ", p->e);
        p = p->next;
    }
}

void main()
{
    int x, pos, ch, i;
    List l, l1;
    l = (struct Node *) malloc(sizeof(struct Node));
    l->previous = NULL;
    l->next = NULL;
    List p = l;
    printf("DOUBLY LINKED LIST IMPLEMENTATION OF LIST ADT\n\n");
    do
    {
        printf("\n\n1. INSERT\t 2. DELETE\t 3. FIND\t 4. PRINT\t 5. QUIT\n\nEnter the choice :: ");
        scanf("%d", &ch);
        switch(ch)
        {
            case 1:
                p = l;
                printf("Enter the element to be inserted :: ");
                scanf("%d",&x);
                printf("Enter the position of the element :: ");
                scanf("%d",&pos);
                for(i = 1; i < pos; i++)
                {
                    p = p->next;
                }
```



```
                Insert(x,l,p);
                break;

            case 2:
                p = l;
                printf("Enter the element to be deleted :: ");
                scanf("%d",&x);
                Delete(x,p);
                break;

            case 3:
                p = l;
                printf("Enter the element to be searched :: ");
                scanf("%d",&x);
                p = Find(x,p);
                if(p == NULL)
                    printf("Element does not exist!!!\n");
                else
                    printf("Element exist!!!\n");
                break;

            case 4:
                Display(l);
                break;
        }
    }
    while(ch<5);
}
```

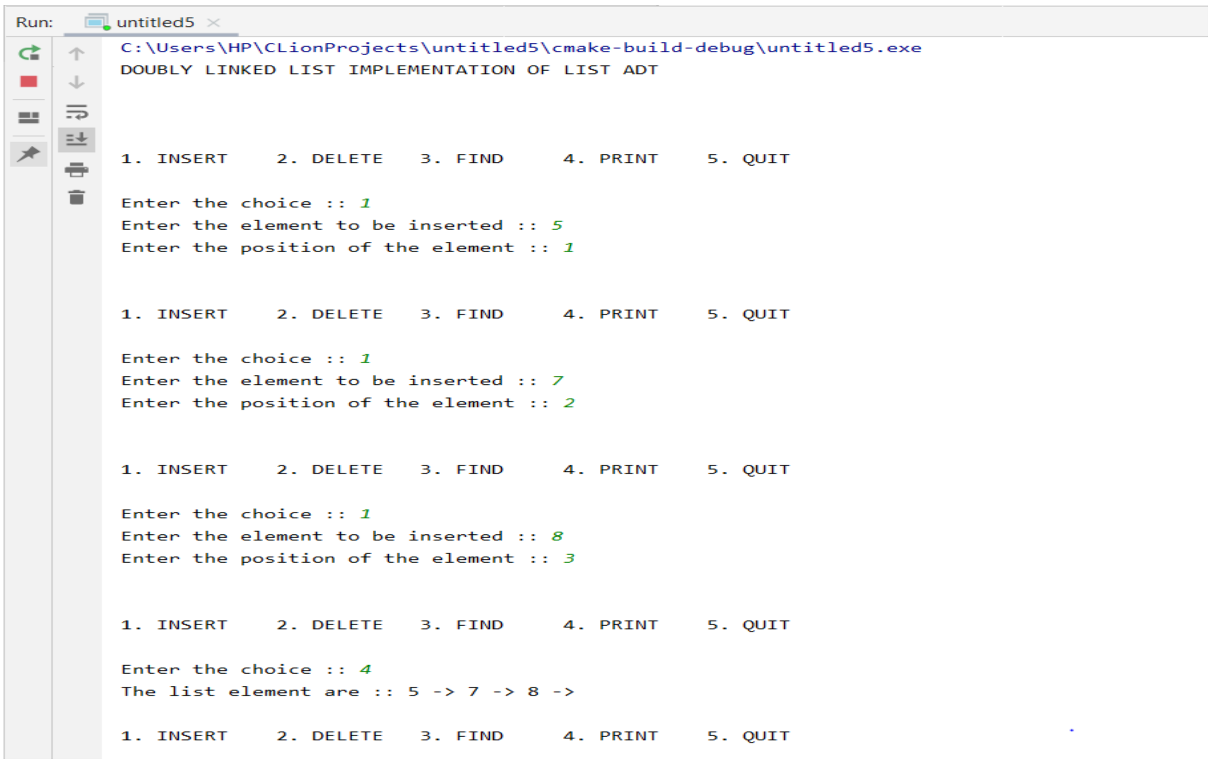
Fig 2 program to create a linked list and delete the element entered by a user.

4. Presentation of Results

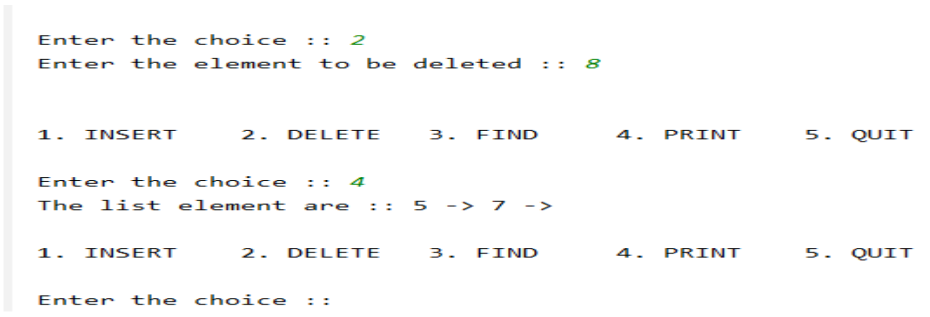


```
C:\Users\HP\CLionProjects\untitled5\cmake-build-debug\untitled5.exe
Enter the no. of nodes to create list: 5
Enter node data for position 1 in the list:  1
Enter node data for position 2 in the list:  2
Enter node data for position 3 in the list:  3
Enter node data for position 4 in the list:  4
Enter node data for position 5 in the list:  5

List elements:
1 2 3 4 5

Insert the element at 1st position:  8

List elements:
8 1 2 3 4 5

Insert the element at last position:  8

List elements:
8 1 2 3 4 5 8

Insert the element at any position in the list
Enter the position: 3
Enter the element value: 2

List elements:
8 1 2 2 3 4 5 8
```

Fig 3 result of program to Implement a linked list operations of insertion of element at end of node and at beginning



```
C:\Users\HP\CLionProjects\untitled5\cmake-build-debug\untitled5.exe
DOUBLY LINKED LIST IMPLEMENTATION OF LIST ADT


1. INSERT    2. DELETE    3. FIND     4. PRINT    5. QUIT

Enter the choice :: 1
Enter the element to be inserted :: 5
Enter the position of the element :: 1


1. INSERT    2. DELETE    3. FIND     4. PRINT    5. QUIT

Enter the choice :: 1
Enter the element to be inserted :: 7
Enter the position of the element :: 2


1. INSERT    2. DELETE    3. FIND     4. PRINT    5. QUIT

Enter the choice :: 1
Enter the element to be inserted :: 8
Enter the position of the element :: 3


1. INSERT    2. DELETE    3. FIND     4. PRINT    5. QUIT

Enter the choice :: 4
The list element are :: 5 -> 7 -> 8 ->

1. INSERT    2. DELETE    3. FIND     4. PRINT    5. QUIT
```



```
Enter the choice :: 2
Enter the element to be deleted :: 8


1. INSERT     2. DELETE    3. FIND      4. PRINT     5. QUIT

Enter the choice :: 4
The list element are :: 5 -> 7 ->

1. INSERT     2. DELETE    3. FIND      4. PRINT     5. QUIT

Enter the choice ::
```

Fig 4 result program to create a linked list and delete the element entered by a user.

5. Conclusions

In this laboratory we learnt to Implement a linked list and Insert a node at the beginning and Insert sa node at the end . In second program we leaent to Print the linked list and Write a program to create a linked list and delete the element entered by a user.