Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Department of Electronics & Telecommunication Engineering
Data Structures & Algorithms Lab (DJ19ECSBL1)

Name: Nihal Nisar Shaikh **SAP ID: 60002200155** **Batch: E1-4**

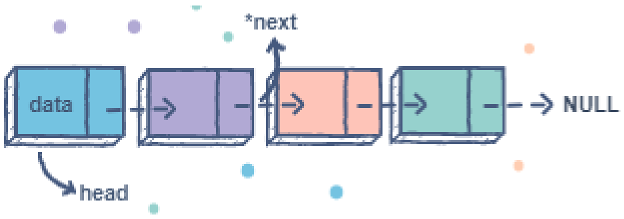
Experiment no: 4 Date:20-10-2022

Aim: Write a program to implement singly linked list.

Programming Language: C/C++/Java

Theory: A singly linked list is a type of linked list that is *unidirectional*, that is, it can be traversed in only one direction from head to the last node (tail). Each element in a linked list is called a node. A single node contains *data* and a pointer to the *next* node which helps in maintaining the structure of the list.

The first node is called the head; it points to the first node of the list and helps us access every other element in the list. The last node, also sometimes called the tail, points to *NULL* which helps us in determining when the list ends.



Declaring a Linked list :

In C language, a linked list can be implemented using structure and pointers.

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

Inserting a new node in a linked list

Case 1: The new node is inserted at the beginning

1. Create and allocate memory for new node
2. Store data
3. Check if it is the first node of the linked list then set next of new node to point to null and set head to point to recently created node
4. If it is not the first node then, set next of new node to point to head and Change head to point to recently created node

Case 2: The new node is inserted at the end

1. Create and allocate memory for new node

2. Store data
3. Check if it is the first node of the linked list then set next of new node to point to null and set head to point to recently created node
4. Else, Traverse to the last node using a temp pointer which initially points to head node, set next field of last node to recently created node and next of new node to point to null value.

Case 3: The new node is inserted after a given node

1. Create and allocate memory for new node
2. Store data
3. Traverse the Linked list upto the node using temp pointer after which the node has to be inserted
4. Set the next pointer of the new node to the next pointer of temp node.
5. Set the next pointer of temp node to next pointer of newnode.

Case 4: The new node is inserted before a given node

1. Create and allocate memory for new node
2. Store data
3. Traverse the linked list using temp pointer upto the node before which the node has to be inserted. Use a q pointer which points to the node which is just behind of temp pointer.
4. Set the next pointer q node to the newnode node.
5. Set the next pointer of newnode to the temp node.

printing/displaying a linked list

1. Define a temp pointer which initially points to the head node. Traverse the linked list using a temp pointer till the temp becomes null and display the data values of the node

Deleting a new node from a linked list

Case 1: The first node is deleted

1. Check if head is pointing to null value it means linked list is empty. Then print "underflow" and exit from the function.
2. Else define a temp pointer, set its value to head pointer. If next field of temp is NULL value it means it is the last node of the linked list. Delete it and free the memory.
3. Else, Shift the head pointer to next node. Delete the node where temp pointer is pointing and then free the memory using free command

Case 2: The last node is deleted

Name: Nihal Nisar Shaikh **SAP ID: 60002200155** **Batch: E1-4**

1. Check if head is pointing to null value it means linked list is empty. Then print "underflow" and exit from the function.

2. Else define a temp pointer, set its value to head pointer. If next field of temp is NULL value it means it is the last node of the linked list. Delete it and free the memory.
3. Else, Traverse through the linked list using temp pointer (increment temp until the last node). Use q pointer which points to the node which is just ahead of the temp node.

4. Set next value of q pointer to null value

4. Delete the node where temp pointer is pointing and then free the memory using free command

Case 3: Delete the node from a given position

1. Check if head is pointing to null value it means linked list is empty. Then print "underflow" and exit from the function.
2. Specify the position of the node to be deleted
3. Use a temp pointer to traverse to the position of the node to be deleted. Use q pointer which points to the node which is just ahead of the temp node
4. Set next value of q pointer to next value of temp pointer
5. Delete the temp node and free the memory

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Department of Electronics & Telecommunication Engineering
Data Structures & Algorithms Lab (DJ19ECSBL1)

Name: Nihal Nisar Shaikh **SAP ID: 60002200155** **Batch: E1-4**

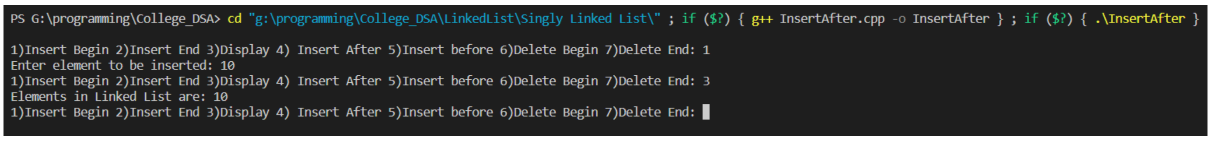
Code:

```
#include <iostream>
#include <malloc.h>
using namespace std;

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

struct node *head = NULL;

void insertbegin()
{
    int value;
    struct node *newnode = NULL;
    newnode = (struct node *)malloc(sizeof(struct node));
    cout << "Enter element to be inserted: ";
    cin >> value;
    newnode->data = value;
    if (head == NULL)
    {
        head = newnode;
        head->next = NULL;
    }
    else
    {
        newnode->next = head;
        head = newnode;
    }
}

void insertAfter()
{
    struct node *newnode;
    newnode = (struct node *)malloc(sizeof(struct node));
    int a;
    cout << "Enter element to be inserted ";
    cin >> a;
    newnode->data = a;
    int num;
    cout << "Enter number after which it should be inserted: ";
    cin >> num;
    struct node *temp;
    temp = head;
    while (temp->data != num)
    {
        temp = temp->next;
    }
    newnode->next = temp->next;
    temp->next = newnode;
}

void insertBefore()
{
    struct node *newnode;
    newnode = (struct node *)malloc(sizeof(struct node));
    int a;
    cout << "Enter element to be inserted ";
    cin >> a;
    newnode->data = a;
    int num;
    cout << "Enter number after which it should be inserted: ";
    cin >> num;
    struct node *temp;
    temp = head;
    struct node *temp1;
    while (temp->data != num)
    {
        temp1 = temp;
        temp = temp->next;
    }
    temp1->next = newnode;
    newnode->next = temp;
}

void deletebegin()
{
    struct node *temp;
    temp = head;
    if (head == NULL)
    {
        cout << "Underflow" << endl;
    }
    else
    {
        if (head->next == NULL)
        {
            head = NULL;
            free(temp);
        }
        else
        {
            head = head->next;
            free(temp);
        }
    }
}

void deleteEnd()
{
    struct node *temp;
    struct node *temp1;
    temp = head;
    if (head == NULL)
    {
        cout << "Underflow" << endl;
    }
    else
    {
        if (head->next == NULL)
        {
            head = NULL;
            free(temp);
        }
        else
        {
            while ((temp->next) != NULL)
            {
                temp1 = temp;
                temp = temp->next;
            }
            free(temp);
            temp1->next = NULL;
        }
    }
}

void deleteposition()
{
    int num;
    cout << "Enter element position to be deleted: ";
    cin >> num;
    struct node *temp;
    struct node *temp1;
    temp = head;
    for (int i = 0; i < num; i++)
    {
        temp1 = temp;
        temp = temp->next;
    }
    temp1->next = temp->next;
    free(temp);
}

void insertend()
{
    struct node *newnode;
    newnode = (struct node *)malloc(sizeof(struct node));
    int value;
    cout << "Enter elemnt to be inserted: ";
    cin >> value;
    newnode->data = value;
    struct node *temp = NULL;
    temp = (struct node *)malloc(sizeof(struct node));
    temp = head;
    if (head == NULL)
    {
        head = newnode;
        head->next = NULL;
        return;
    }
    while ((temp->next) != NULL)
    {
        temp = temp->next;
    }
    temp->next = newnode;
    newnode->next = NULL;
    cout << "Value inserted is: " << newnode->data << endl;
}

void display()
{
    if (head == NULL)
    {
        cout << "Linked List is empty: " << endl;
        return;
    }
    struct node *temp = NULL;
    temp = head;
    cout << "Elements in Linked List are: ";
    while (temp != NULL)
    {
        cout << temp->data << " ";
        temp = temp->next;
    }
    cout << endl;
}

int main()
{
    while (1)
    {
        int choice;
        cout << "1)Insert Begin 2)Insert End 3)Display 4) Insert After
5)Insert before 6)Delete Begin 7)Delete End 8)Delete position: ";
        cin >> choice;
        switch (choice)
        {
        case 1:
            insertbegin();
            break;

        case 2:
            insertend();
            break;
        case 3:
            display();
            break;
        case 4:
            insertAfter();
            break;
        case 5:
            insertBefore();
            break;
        case 6:
            deletebegin();
            break;
        case 7:
            deleteEnd();
            break;
        case 8:
            deleteposition();
            break;
        default:
            break;
        }
    }
    return 0;
}
```

Name: Nihal Nisar Shaikh **SAP ID: 60002200155** **Batch: E1-4**

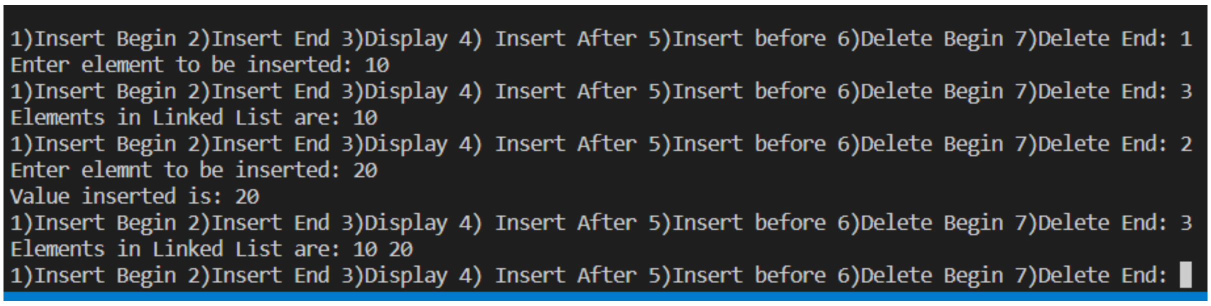
Output:

The new node is inserted at the beginning



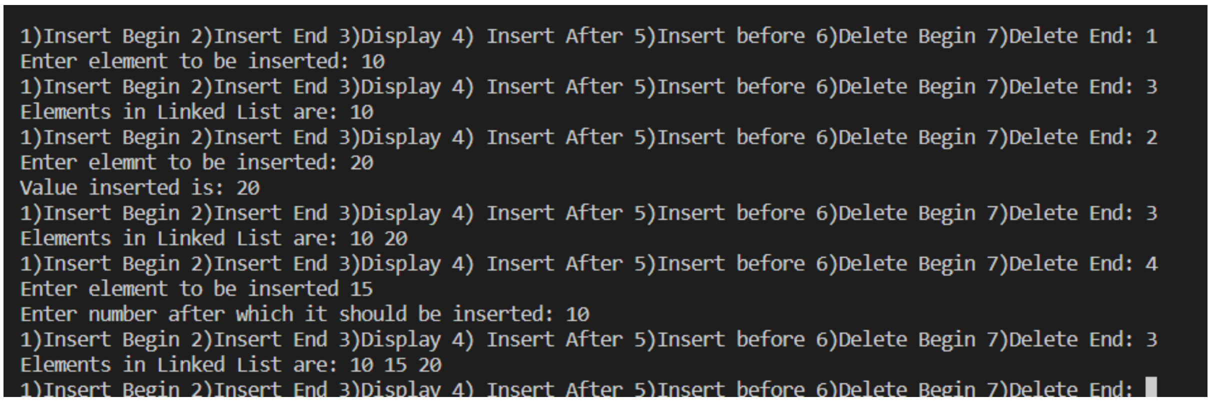
```
PS G:\programming\College_DSA> cd "g:\programming\College_DSA\LinkedList\Singly Linked List\" ; if ($?) { g++ InsertAfter.cpp -o InsertAfter } ; if ($?) { .\InsertAfter }

1)Insert Begin 2)Insert End 3)Display 4) Insert After 5)Insert before 6)Delete Begin 7)Delete End: 1
Enter element to be inserted: 10
1)Insert Begin 2)Insert End 3)Display 4) Insert After 5)Insert before 6)Delete Begin 7)Delete End: 3
Elements in Linked List are: 10
1)Insert Begin 2)Insert End 3)Display 4) Insert After 5)Insert before 6)Delete Begin 7)Delete End: 
```

The new node is inserted at the end:



```
1)Insert Begin 2)Insert End 3)Display 4) Insert After 5)Insert before 6)Delete Begin 7)Delete End: 1
Enter element to be inserted: 10
1)Insert Begin 2)Insert End 3)Display 4) Insert After 5)Insert before 6)Delete Begin 7)Delete End: 3
Elements in Linked List are: 10
1)Insert Begin 2)Insert End 3)Display 4) Insert After 5)Insert before 6)Delete Begin 7)Delete End: 2
Enter elemnt to be inserted: 20
Value inserted is: 20
1)Insert Begin 2)Insert End 3)Display 4) Insert After 5)Insert before 6)Delete Begin 7)Delete End: 3
Elements in Linked List are: 10 20
1)Insert Begin 2)Insert End 3)Display 4) Insert After 5)Insert before 6)Delete Begin 7)Delete End: 
```

The new node is inserted after a given node

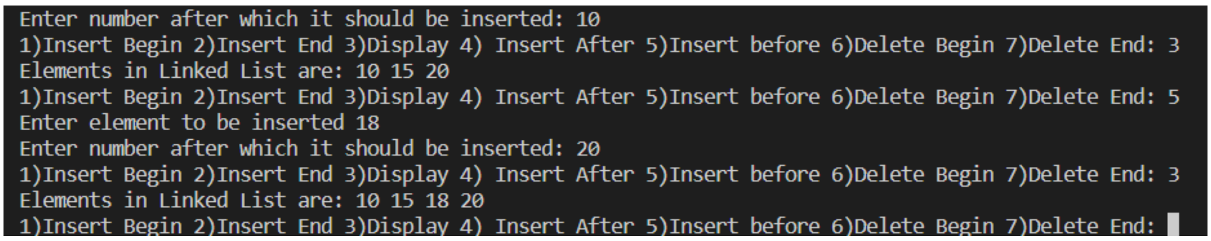


```
1)Insert Begin 2)Insert End 3)Display 4) Insert After 5)Insert before 6)Delete Begin 7)Delete End: 1
Enter element to be inserted: 10
1)Insert Begin 2)Insert End 3)Display 4) Insert After 5)Insert before 6)Delete Begin 7)Delete End: 3
Elements in Linked List are: 10
1)Insert Begin 2)Insert End 3)Display 4) Insert After 5)Insert before 6)Delete Begin 7)Delete End: 2
Enter elemnt to be inserted: 20
Value inserted is: 20
1)Insert Begin 2)Insert End 3)Display 4) Insert After 5)Insert before 6)Delete Begin 7)Delete End: 3
Elements in Linked List are: 10 20
1)Insert Begin 2)Insert End 3)Display 4) Insert After 5)Insert before 6)Delete Begin 7)Delete End: 4
Enter element to be inserted 15
Enter number after which it should be inserted: 10
1)Insert Begin 2)Insert End 3)Display 4) Insert After 5)Insert before 6)Delete Begin 7)Delete End: 3
Elements in Linked List are: 10 15 20
1)Insert Begin 2)Insert End 3)Display 4) Insert After 5)Insert before 6)Delete Begin 7)Delete End: 
```

Department of Electronics & Telecommunication Engineering
Data Structures & Algorithms Lab (DJ19ECSBL1)

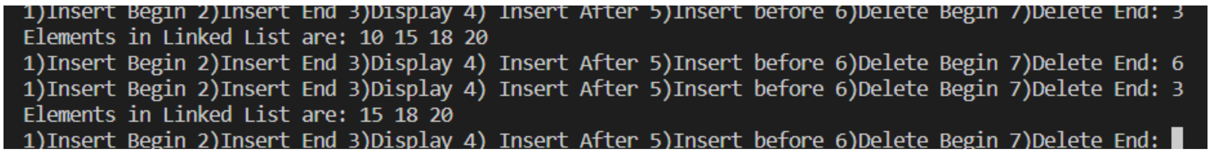
Name: Nihal Nisar Shaikh **SAP ID: 60002200155** **Batch: E1-4**

The new node is inserted before a given node



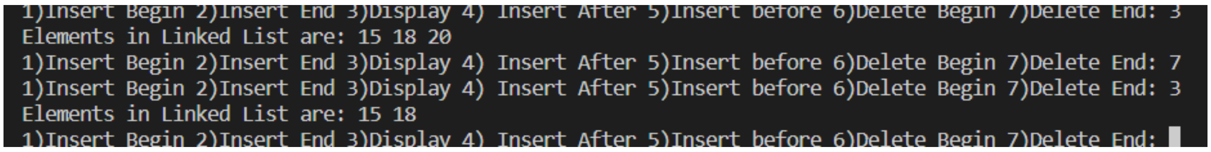
```
Enter number after which it should be inserted: 10
1)Insert Begin 2)Insert End 3)Display 4) Insert After 5)Insert before 6)Delete Begin 7)Delete End: 3
Elements in Linked List are: 10 15 20
1)Insert Begin 2)Insert End 3)Display 4) Insert After 5)Insert before 6)Delete Begin 7)Delete End: 5
Enter element to be inserted 18
Enter number after which it should be inserted: 20
1)Insert Begin 2)Insert End 3)Display 4) Insert After 5)Insert before 6)Delete Begin 7)Delete End: 3
Elements in Linked List are: 10 15 18 20
1)Insert Begin 2)Insert End 3)Display 4) Insert After 5)Insert before 6)Delete Begin 7)Delete End:
```

The first node is deleted



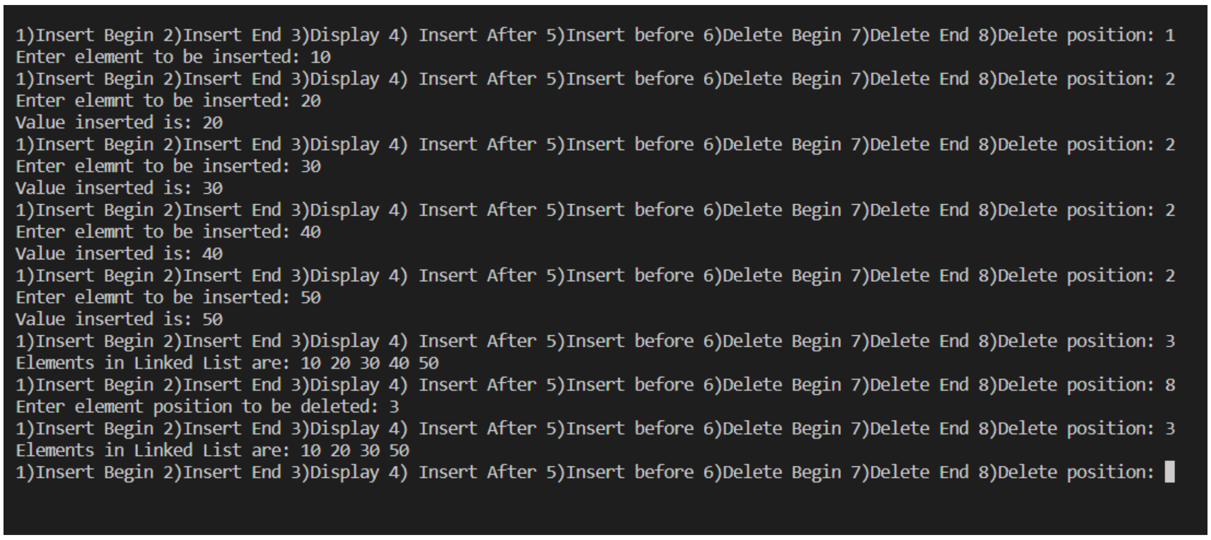
```
1)Insert Begin 2)Insert End 3)Display 4) Insert After 5)Insert before 6)Delete Begin 7)Delete End: 3
Elements in Linked List are: 10 15 18 20
1)Insert Begin 2)Insert End 3)Display 4) Insert After 5)Insert before 6)Delete Begin 7)Delete End: 6
1)Insert Begin 2)Insert End 3)Display 4) Insert After 5)Insert before 6)Delete Begin 7)Delete End: 3
Elements in Linked List are: 15 18 20
1)Insert Begin 2)Insert End 3)Display 4) Insert After 5)Insert before 6)Delete Begin 7)Delete End:
```

The last node is deleted



```
1)Insert Begin 2)Insert End 3)Display 4) Insert After 5)Insert before 6)Delete Begin 7)Delete End: 3
Elements in Linked List are: 15 18 20
1)Insert Begin 2)Insert End 3)Display 4) Insert After 5)Insert before 6)Delete Begin 7)Delete End: 7
1)Insert Begin 2)Insert End 3)Display 4) Insert After 5)Insert before 6)Delete Begin 7)Delete End: 3
Elements in Linked List are: 15 18
1)Insert Begin 2)Insert End 3)Display 4) Insert After 5)Insert before 6)Delete Begin 7)Delete End:
```

Delete the node from a given position



```
1)Insert Begin 2)Insert End 3)Display 4) Insert After 5)Insert before 6)Delete Begin 7)Delete End 8)Delete position: 1
Enter element to be inserted: 10
1)Insert Begin 2)Insert End 3)Display 4) Insert After 5)Insert before 6)Delete Begin 7)Delete End 8)Delete position: 2
Enter elemnt to be inserted: 20
Value inserted is: 20
1)Insert Begin 2)Insert End 3)Display 4) Insert After 5)Insert before 6)Delete Begin 7)Delete End 8)Delete position: 2
Enter elemnt to be inserted: 30
Value inserted is: 30
1)Insert Begin 2)Insert End 3)Display 4) Insert After 5)Insert before 6)Delete Begin 7)Delete End 8)Delete position: 2
Enter elemnt to be inserted: 40
Value inserted is: 40
1)Insert Begin 2)Insert End 3)Display 4) Insert After 5)Insert before 6)Delete Begin 7)Delete End 8)Delete position: 2
Enter elemnt to be inserted: 50
Value inserted is: 50
1)Insert Begin 2)Insert End 3)Display 4) Insert After 5)Insert before 6)Delete Begin 7)Delete End 8)Delete position: 3
Elements in Linked List are: 10 20 30 40 50
1)Insert Begin 2)Insert End 3)Display 4) Insert After 5)Insert before 6)Delete Begin 7)Delete End 8)Delete position: 8
Enter element position to be deleted: 3
1)Insert Begin 2)Insert End 3)Display 4) Insert After 5)Insert before 6)Delete Begin 7)Delete End 8)Delete position: 3
Elements in Linked List are: 10 20 30 50
1)Insert Begin 2)Insert End 3)Display 4) Insert After 5)Insert before 6)Delete Begin 7)Delete End 8)Delete position:
```

Department of Electronics & Telecommunication Engineering
Data Structures & Algorithms Lab (DJ19ECSBL1)

Name: Nihal Nisar Shaikh **SAP ID: 60002200155** **Batch: E1-4**

Result and Conclusion:

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers which points to their next element.

1) Inserted Node:
   a. At beginning.
   b. At end.
   c. After particular node.
2) Deleted Node:
   a. At beginning.
   b. At end.
   c. Present at particular position.
3) Displayed element in linked list.