THE UNIVERSITY
OF LAHORE
ISLAMABAD
CAMPUS

Data Structures & Algorithms
(CS09203)

Lab Report

Name:	Muhammad Umer
Registration #:	CSU-F16-104
Lab Report #:	05
Dated:	30-04-2018
Submitted To:	Mr. Usman Ahmed

The University of Lahore, Islamabad Campus
Department of Computer Science & Information Technology

Experiment # 1
Link List-Basic Insertion, Deletion and Traversal

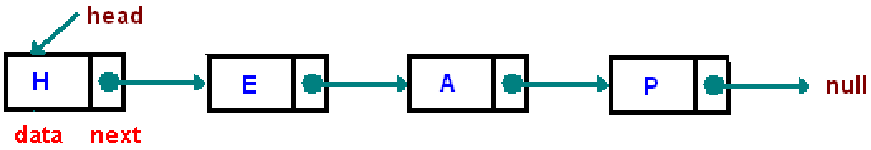
Objective
To understand and implement the Link List with basic Insertion, Deletion at desired position and Traversal.

Software Tool
1. Sublime Text Editor
2. Dev C++
3. Window 7 (32 Bit)

1 Theory

One disadvantage of using arrays to store data is that arrays are static structures and therefore cannot be easily extended or reduced to fit the data set. Arrays are also expensive to maintain new insertions and deletions. In this chapter we consider another data structure called Linked Lists that addresses some of the limitations of arrays.

A linked list is a linear data structure where each element is a separate object.



Each element (we will call it a node) of a list is comprising of two items - the data and a reference to the next node. The last node has a reference to null. The entry point into a linked list is called the head of the list. It should be noted that head is not a separate node, but the reference to the first node. If the list is empty then the head is a null reference.

A linked list is a dynamic data structure. The number of nodes in a list is not fixed and can grow and shrink on demand. Any application which has to deal with an unknown number of objects will need to use a linked list.

One disadvantage of a linked list against an array is that it does not allow direct access to the individual elements. If you want to access a particular item then you have to start at the head and follow the references until you get to that item.

2 Task

2.1 Procedure: Task 1 Insertion

In this Linked List user can insert integer type of the data and the data will always be inserted in the start of the list.

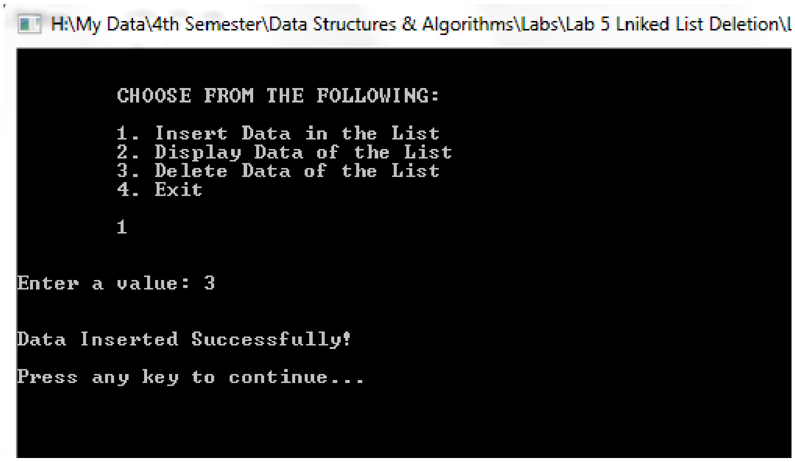
```
void insert(int item){
        node *NewNode = (node*) malloc(sizeof(node));
        NewNode -> data = item;
        NewNode -> next = head;
        head = NewNode;
}

Please see Figure:1 on page 4 for insertion
```

2.2 Procedure: Task 2 Delete

```
void Delete(int n){
        if(head == NULL){
                cout<<"\n\nError:␣Empty␣List!";
                return;
        }
        node* NewNode = (node*)malloc(sizeof(node));
        if(n == 1){
                NewNode = head;
                head = NewNode -> next;
                free(NewNode);
                return;
        }
        else{
        NewNode = head;
        for(int i=0; i<n-2; i++)
                NewNode = NewNode -> next;
        node* NewNode2 = (node*)malloc(sizeof(node));
        NewNode2 = NewNode -> next;
        NewNode -> next = NewNode2 -> next;
        free(NewNode2);
        return;
        }
}

Please see Figure:3 on page 5 for Deletion
```

2.3 Procedure: Task 2 Traverse

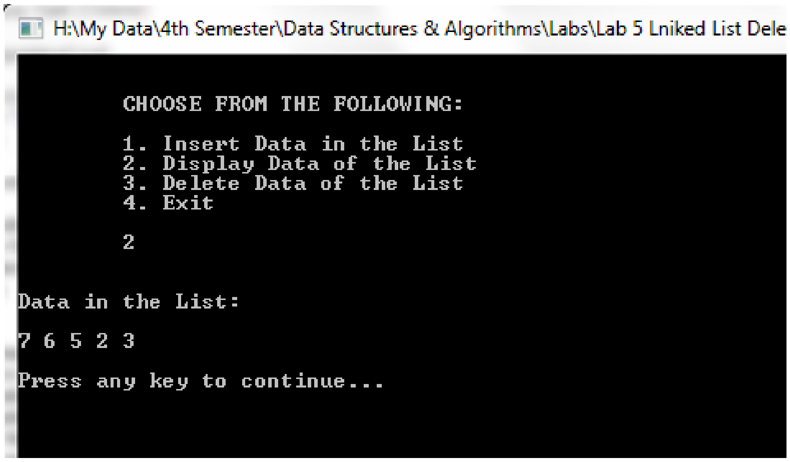
```
void display(){
        if(head == NULL){
                cout<<”\n\nError:␣Empty␣List!”;
                cout<<”\n\nPress␣any␣key␣to␣continue...”;
                getch();
                return;
        }
        node *NewNode = (node*) malloc(sizeof(node));
        NewNode = head;
        cout<<”\n\nData␣in␣the␣List:\n\n”;
        while(NewNode != NULL){
                cout<<NewNode -> data<<”␣”;
                NewNode = NewNode -> next;
        }
}

Please see Figure:2 on page 4 for Traversing
```



H:\My Data\4th Semester\Data Structures & Algorithms\Labs\Lab 5 Lniked List Deletion\l

```
        CHOOSE FROM THE FOLLOWING:

        1. Insert Data in the List
        2. Display Data of the List
        3. Delete Data of the List
        4. Exit

        1


Enter a value: 3


Data Inserted Successfully!

Press any key to continue...
```

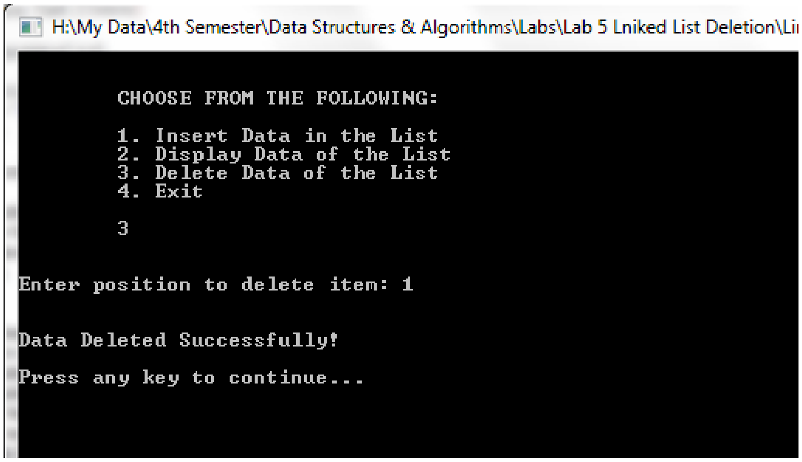
Figure 1: Inserting in the list



H:\My Data\4th Semester\Data Structures & Algorithms\Labs\Lab 5 Lniked List Dele

```
        CHOOSE FROM THE FOLLOWING:

        1. Insert Data in the List
        2. Display Data of the List
        3. Delete Data of the List
        4. Exit

        2


Data in the List:

7 6 5 2 3

Press any key to continue...
```

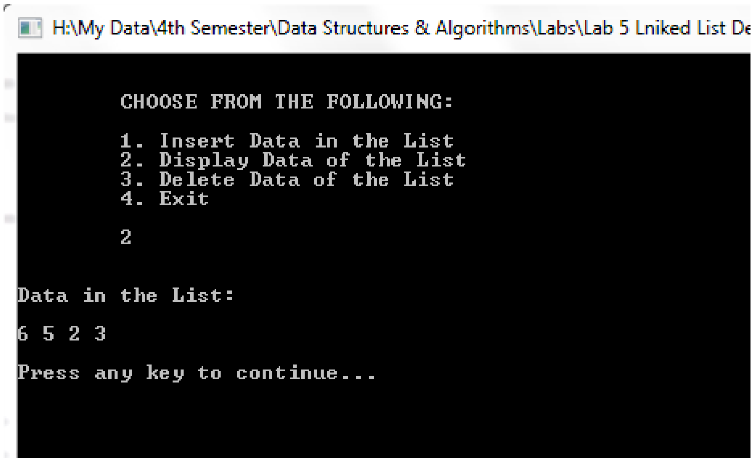
Figure 2: Displaying After insertion



H:\My Data\4th Semester\Data Structures & Algorithms\Labs\Lab 5 Lniked List Deletion\Li

```
        CHOOSE FROM THE FOLLOWING:

        1. Insert Data in the List
        2. Display Data of the List
        3. Delete Data of the List
        4. Exit

        3


Enter position to delete item: 1


Data Deleted Successfully!

Press any key to continue...
```

Figure 3: Deleting item from the list



H:\My Data\4th Semester\Data Structures & Algorithms\Labs\Lab 5 Lniked List De

```
        CHOOSE FROM THE FOLLOWING:

        1. Insert Data in the List
        2. Display Data of the List
        3. Delete Data of the List
        4. Exit

        2


Data in the List:

6 5 2 3

Press any key to continue...
```

Figure 4: Displaying after insertion

Source Code
https://github.com/umerayan/Data-Structure-and-Algorithms

3 Conclusion

A linked list is a linear data structure where each element is a separate object. Each element is called as a node, that contains two item - the data and a reference to the next node. The last node has a reference to null. The entry point into a linked list is called the head of the list. A linked list is a dynamic data structure. The number of nodes in a list is not fixed and can grow and shrink on demand.
We perform Insertion, Traverser and Deletion at desired position in this experiment, you can find the link for the source code given above.

____

(Concerned Teacher/Lab Engineer)