**National University of Computer and Emerging Sciences**



**Laboratory Manual**

*for*

# Data Structures Lab

|  |  |
| --- | --- |
| Course Instructor | Mr. Uzair Naqvi |
| Lab Instructor | Mr. Durraiz Waseem |
| Section | BCS-3H |
| Date | Sept 12, 2024 |
| Semester | Fall 2024 |

# Department of Computer Science

FAST-NU, Lahore, Pakistan

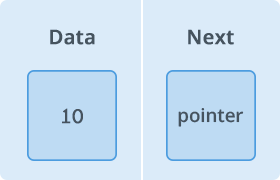
**Objectives:**

In this lab, students will practice singly link list implementation and other basic operations on it like insertion and removal of data elements from it

###### 

###### **Linked List Overview:**

A linked list is a way to store a collection of elements. Like an array these can be character or integers. Each element in a linked list is stored in the form of a node. A linked list is formed when many such nodes are linked together to form a chain.



**Figure 1. Node**

It 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 as shown in the below image:



**Figure 2. Linked list**

✔ List is a collection of components, called nodes. Every node (except the last node) contains the address of the next node. Every node in a linked list has two components:

* One to store the relevant information that is, **data**
* One to store the address, called the link or **next**, of the next node in the list.

✔ The address of the first node in the list is stored in a separate location, called the **head** or first.

✔ The address of the last node in the list is stored in a separate location, called the **tail** or last.

There are 3 types of linked list:

* Singly Linked List
* Circular Linked List
* Doubly Linked List

In this lab we will discuss singly linked list.

**Part-1:** Implement a Struct ‘*Node’* that contains two data members: An int variable ‘*data’* and Node pointer ‘*next’*.

**Part-2:** Now implement a simple linked list class having two private data member Node pointer ‘*head’* and Node pointer ‘*tail’*. Please note that Node class is a nested class of linked list class. (Note that Struct Node is defined inside the List class)

**Part-3:** Now implement the following operations for linked list class:

a. Insert at start, void insertAtHead(T const element);

b. Insert at end, void insertAtTail(T const element);

c. Update the data element at specific position, void update( Kth position).

d. Search the key, bool search(T const element);

e. Delete key, bool delete( T const element);

**f. Get the size of the list**: int size();

g. Insert at the middle using above size function, void insertAtMiddle( T const element);

h. **Reverse the list**: void reverse(); It should actually change nodes instead of just data.

**Part-4:** Now implement the following operations for linked list class:

a. Print void print() const;

b. Delete at Start, void eraseAtHead ();

c. Delete at End, void eraseAtTail();

d. deleteInRange: Delete all nodes that lie in given range. For example if range is (2, 4) then deletion starts from 2nd node of list and ends at 4th node.

e. Destructor

**Part-3 operations explained:-**



**15**

**5**

**21**

**20**

**Start=10**

**Next pointer location=11**

**10**

**11**

**12**

**13**

**11**

**12**

**13**

**NULL**

**Figure 3. Singly Linked list**

* 1. Function called **InsertAtHead** to add node at the begging of list.

After insertion of value ‘**12**’ at the beginning of the linked list, it will become as following:



**Start=9**

**9**

**10**

**11**

**12**

**13**

**12**

**10**

**11**

**12**

**13**

**NULL**



5

15

21

20

**Figure 3a. Insertion at beginning of singly linked list**

* 1. Function called **InsertAtTail** to add node at the last of list.

After insertion of value ‘**25**’ at the end of the linked list, it will become as following:

**Start=10**

**Next pointer location=11**

**11**

**12**

**13**

**14**

**NULL**

**11**

**12**

**13**

**14**

5

20

21

**25**

**10**

15

**Figure 3b. Insertion at the end of the singly linked list**

* 1. Function called update to **update** the value at kth position.

**Start=10**

**Next pointer location=11**

**10 11 12 13**

**12**

**20**

5

**13**

21

15 **11**

**NULL**

If we want to update the value ‘**20**’ which is residing at location x12 (k = 2 position) to ‘**25**’, our linked list will display as following:   
  
 15->5->**25**->21-> null  
 **Figure 3c. Updating at the specific index of singly linked list**