**LAB REPORT ON**

**Data Structures**



**Lab No. 3**

**Date 2020/09/25**

**Topic: Linked List**

**Submitted By**  **Submitted To**

Tapendra Pandey

BCT-D Department of Electronics

075BCT093 and Computer Engineering

**INTRODUCTION**

**Linked List:**

A linked list is a linear data structure where each element is a separate object. Each element (called node) of a list comprises of two items – data and pointer to next node in list. The first node in the list is called head and last node points to null.

There are two types of implementation of lists: static and dynamic implementation.

**Singly Linked List**

Singly linked lists contain nodes, which have a data part as well as an address part i.e. next, which points to the next node in the sequence of nodes.

The operations we can perform on singly linked lists are insertion, deletion and traversal.

**Doubly Linked List**

In a doubly linked list, each node contains a data part and two addresses, one for the previous node and one for the next node.

**Circular Linked List**

In circular linked list, the last node of the list holds the address of the first node hence forming a circular chain.

**Application of linked lists:**

1. Implementation of stacks and queues

2. Implementation of graphs

3. Dynamic memory allocation

4. Maintaining directory of names

5. Performing arithmetic operations on long integers

**ALGORITHMS**

Algorithm for following cases in **singly linear linked list**:

1. **insertion operations**
   1. insert node at beginning of the list
   2. insert node at end of the list
   3. insert node after specific node
   4. insert node before specific node
2. **deletion operations**
   1. delete node from beginning of the list
   2. delete node from end of the list
   3. delete node after specific node

**a) insert node at the beginning of the list**

step 1: if avail = null

write overflow

go to step 7

[end of if]

step 2: set new node = avail

step 3: set avail = avail -> next

step 4: set new node -> data val

step 5: set new node -> next = start

step 6: set start = new node

step 7: exit

**b) insert node at end of the list**

step 1: if avail = null

write overflow

go to step 10

[end of if]

step 2: set new node= avail

step 3: set avail= avail -> next

step 4: set new node- > data = val

step 5: set new node - > next = null

step 6: set ptr = start

step 7: repeat step 8 while ptr -> next != null

step 8: set ptr = ptr -> next

[end of loop)

step 9: set ptr -> next = new\_node

step 10: exit

**c) insert node after specific node**

step 1: if avail = null

write overflow

go to step 12

[end of if]

step 2: set new node = avail

step 3. set avail = avail-> next = val

step 4: set new node -> data

step 5: set ptr = start

step 6: set preptr = ptr

step 7: repeat steps 8 and 9 while preptr-> data 1= num

step 8:

set preptr = ptr

step 9: set ptr = ptr -> next

[end of loop)

step 10: preptr-> next = new\_node

step 11: set new node -> next = ptr

step 12: extt

**d) insert node before specific node**

step 1: if avail null

write overflow

go to step 12

[end of if]

step 2: set new node = avail

step 3: set avail = avail -> next

step 4: set new node-> data = val

step 5: set ptr =start

step 6: set preptr = ptr

step 7: repeat steps 8 and 9 while ptr-> data != num

step 8: set preptr = ptr

step 9: set ptr - ptr -> next

[end of loop)

step 10: preptr -> next= new node

step 11: set new node -> next =ptr

step 12: exit

**2.**

**a) Delete node from beginning of the list**

step 1: if start = null

write underflow

go to step 5

[end of if]

step 2: set ptr = start

step 3: set start = start -> next

step 4: free ptr

step 5: exit

**b) Delete from the end**

step 1: if start = null

write underflow

go to step 8

[end of if]

step 2: set ptr = start

step 3: repeat steps 4 and 5 while ptr> next !=

step 4: set preptr = ptr

step 5: set ptr ptr -> next

[end of loop]

step 6: set ptr ptr -> next = null

step 7: free ptr

step 8: exit

**c) Delete after specific node**

step 1: if start = null

write underflow

go to step 10

[end of if ]

step 2: set ptr=start

step 3: set preptr = ptr

step 4: repeat steps 5 and 6 while preptr->data != num

step 5: set preptr = ptr

step 6:

set ptr = ptr -> next

[end of loop)

step 7: set temp = ptr

step 8: set ptr ptr -> next = ptr -> next

step 9: free temp

step 10: exit

**DISCUSSION AND CONCLUSION**

In this lab, we learnt about the linked list and its operations. We did operations as insertion of node at beginning of the list, end of the list, after specific node andbefore specific node. Also deletion operations were done from beginning of the list, end of the list and after specific node. The dynamic and static implementations of linked list were done.

Hence, in the lab we understood and implemented algorithms of list datatype.