Practical # 06

Objective:

Discuss *and Analyze implementation of Doubly Linked List Data Structure. Write a C++ program to create a Doubly linked list of integers.*

Theory:

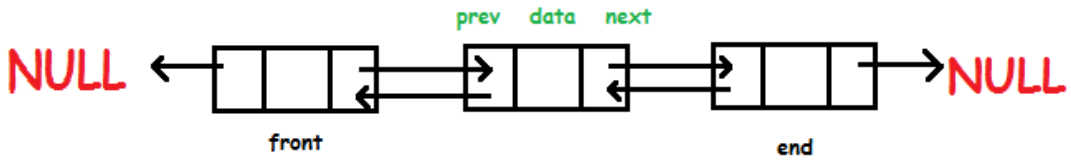
In this Lab, we will introduce the doubly Linked list and possible operations performed on doubly linked list.

Lab Objectives:

- To be able to declare a doubly linked list.
- To be able to perform fundamental operations on doubly linked list.

Introduction:

Doubly linked list is a type of linked list in which each node apart from storing its data has two links. The first link points to the previous node in the list and the second link points to the next node in the list. The first node of the list has its previous link pointing to NULL similarly the last node of the list has its next node pointing to NULL. The two links help us to traverse the list in both backward and forward direction. But storing an extra link requires some extra space.



Implementation a Node type

First we define the node.

```
struct node
{       int data;           // Data
        node *prev;         // A reference to the previous node
        node *next;         // A reference to the next node
};
```

Basic Operations on doubly linked list

Following are some of the operations that we can perform on a doubly linked list.

a) Insertion
Insertion operation of the doubly linked list inserts a new node in the linked list. Depending on the position where the new node is to be inserted, we can have the following insert operations.

- **Insertion at front –** Inserts a new node as the first node.
- **Insertion at the end –** Inserts a new node at the end as the last node.
- **Insertion before a node –** Given a node, inserts a new node before this node.
- **Insertion after a node –** Given a node, inserts a new node after this node.

b) Deletion
Deletion operation deletes a node from a given position in the doubly linked list.

- **Deletion of the first node –** Deletes the first node in the list
- **Deletion of the last node –** Deletes the last node in the list.
- **Deletion of a node given the data –** Given the data, the operation matches the data with the node data in the linked list and deletes that node.

c) Traversal
Traversal is a technique of visiting each node in the linked list. In a doubly linked list, we have two types of traversals as we have two pointers with different directions in the doubly linked list.

- **Forward traversal –** Traversal is done using the next pointer which is in the forward direction.
- **Backward traversal –** Traversal is done using the previous pointer which is the backward direction.

d) Reverse
This operation reverses the nodes in the doubly linked list so that the first node becomes the last node while the last node becomes the first node.

e) Search
Search operation in the doubly linked list is used to search for a particular node in the linked list. For this purpose, we need to traverse the list until a matching data is found.

Example program: *Create and Traverse doubly linked list.*

```
#include <iostream>
using namespace std;

//node structure
struct Node {
    int data;
    Node* next;
    Node* prev;
};

class LinkedList {
  public:
    Node* head;
  public:
    //constructor to create an empty LinkedList
    LinkedList(){
      head = NULL;
    }

    //display the content of the list
    void PrintList() {
      Node* temp = head;
      if(temp != NULL) {
        cout<<"The list contains: ";
        while(temp != NULL) {
          cout<<temp->data<<" ";
          temp = temp->next;
        }
        cout<<endl;
      } else {
        cout<<"The list is empty.\n";
      }
    }
};

// test the code
int main() {
  //create an empty LinkedList
  LinkedList MyList;

  //Add first node.
  Node* first = new Node();
  first->data = 10;
  first->next = NULL;
  first->prev = NULL;
  //linking with head node
  MyList.head = first;

  //Add second node.
  Node* second = new Node();
  second->data = 20;
  second->next = NULL;
  //linking with first node
  second->prev = first;
  first->next = second;

  //Add third node.
  Node* third = new Node();
  third->data = 30;
  third->next = NULL;
  //linking with second node
  third->prev = second;
  second->next = third;

  //print the content of list
  MyList.PrintList();
  return 0;
}
```

OUTPUT

```
The list contains: 10 20 30
```

Review Questions/ Exercise:

1. Write a program to insert a new node at the beginning of doubly linked list.

2. Write a program to insert a new node before given node in doubly linked list.

3. Write a program to insert new node at the end of doubly linked list.

4. Write a program to delete node from doubly linked list.

5. Write a program to **show the Reverse Doubly Linked List.**

Name: _____

Roll# _____

Date: _____

Subject Teacher

Remarks: