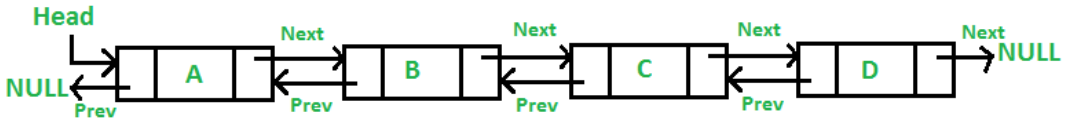
Data Structures Lab 6

Course: Data Structures (CL2001) **Semester:** Fall 2022
Instructor: Muhammad Monis **T.A:** N/A

Note:

- Maintain discipline during the lab.
- Listen and follow the instructions as they are given.
- Just raise hand if you have any problem.
- Completing all tasks of each lab is compulsory.
- Get your lab checked at the end of the session.

Doubly Link List



Doubly Linked List is a variation of Linked list in which navigation is possible in both ways, either forward or backward easily as compared to Single Linked List.

- **Link** − each link of a linked list can store a data called an element.
- **Next** − each link of a linked list contains a link to the next link called Next.
- **Prev** − each link of a linked list contains a link to the previous link called Prev.

```
class Node {
    public:
        int key;
        int data;
        Node * next;
        Node * previous;

         Node() {
            key = 0;
            data = 0;
            next = NULL;
            previous = NULL;
        }

        Node(int k, int d) {
            key = k;
            data = d;
```

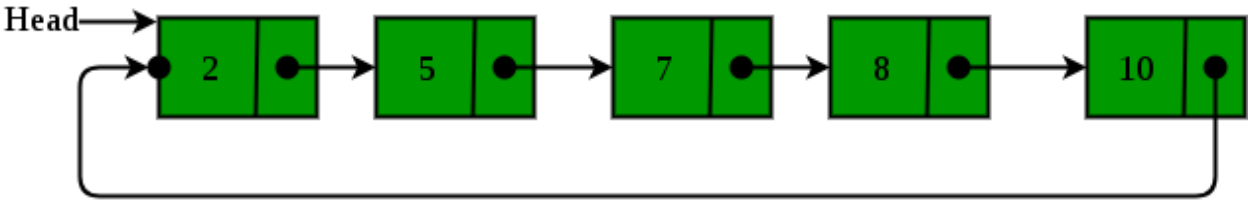
```
        }
};

class DoublyLinkedList {
    public:
        Node * head;

        DoublyLinkedList() {
            head = NULL;
        }

        DoublyLinkedList(Node * n) {
            head = n;
    }

    appendNode();
    prependNode();
    insertNodeAfter();
    deleteNodeByKey();
    updateNodeByKey();
};
```

Task-1:

Create a doubly link list and perform the mentioned tasks.

i. Insert a new node at the end of the list.
ii. Insert a new node at the beginning of list.
iii. Insert a new node at given position.
iv. Delete any node.
v. Print the complete doubly link list.

Circular Link List



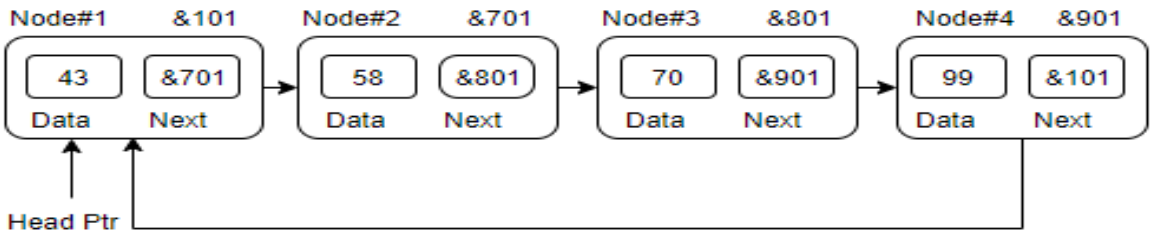
*The **circular linked list** is a linked list where all nodes are connected to form a circle. In a circular linked list, the first node and the last node are connected to each other which forms a circle. There is no NULL at the end.*

```
class Node {
        public:
                int key;
                int data;
                Node * next;

                Node() {
                        key = 0;
                        data = 0;
                        next = NULL;
                }

                Node(int k, int d) {
                        key = k;
                        data = d;
                }
};

class CircularLinkedList {
        public:
                Node * head;

                CircularLinkedList() {
                        head = NULL;
                }


        appendNode();
        prependNode();
        insertNodeAfter();
        deleteNodeByKey();
        updateNodeByKey();
        print();
};
```

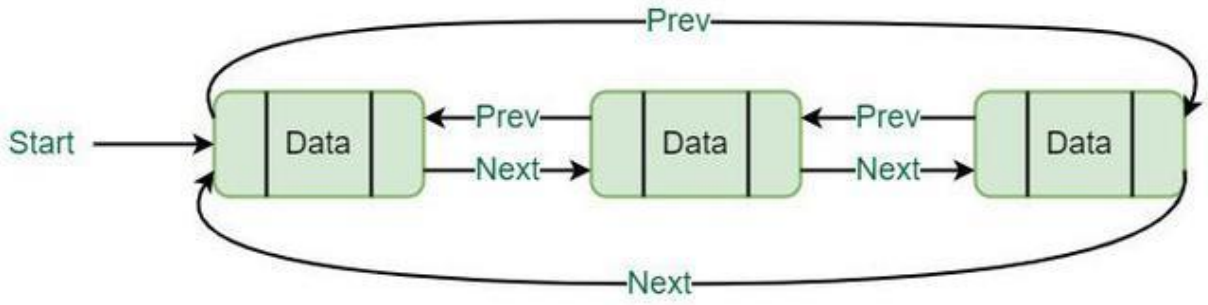
Task-2:

Create a circular link list and perform the mentioned tasks.

i. Insert a new node at the end of the list.
ii. Insert a new node at the beginning of list.
iii. Insert a new node at given position.
iv. Delete any node.
v. Print the complete circular link list.



Circular Double Link List



In doubly linked list, the next pointer of the last node points to the first node and the previous pointer of the first node points to the last node making the circular in both directions.

- The last link's next points to the first link of the list in both cases of singly as well as doubly linked list.
- The first link's previous points to the last of the list in case of doubly linked list.

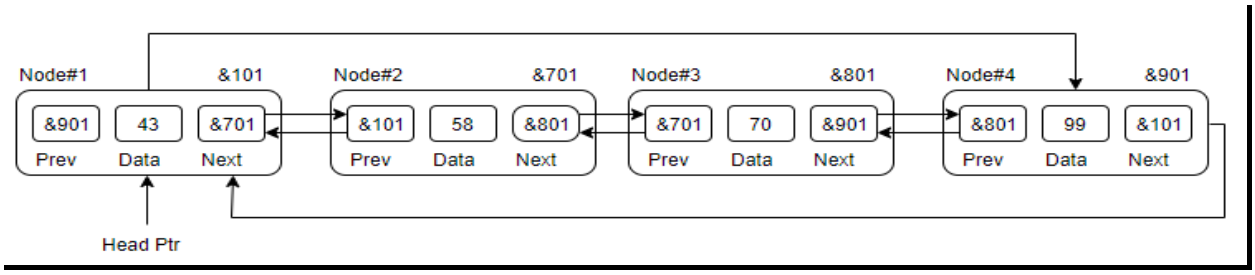
```
class  node
{
public:
        int info;
        node *next;
        node *prev;
};
```

```
class double_clist
{
        public:
        node *create_node(int);

        insert_begin();
        insert_last();
        insert_pos();
        delete_pos();
         update();
        display();
        node  *start, *last;

        double_clist()
        {
                start = NULL;
                last = NULL;
        }
};
```

Task-3:

Create a circular Double link list and perform the mentioned tasks.

i. Insert a new node at the end of the list.
ii. Insert a new node at the beginning of list.
iii. Insert a new node at given position.
iv. Delete any node.
v. Print the complete circular double link list.



Task-4:

Give an efficient algorithm for concatenating two doubly linked lists **L** and **M**, with head and tail preserved nodes, into a single list that contains all the nodes of **L** followed by all the nodes of **M**.