Data Structures Lab 03(b)

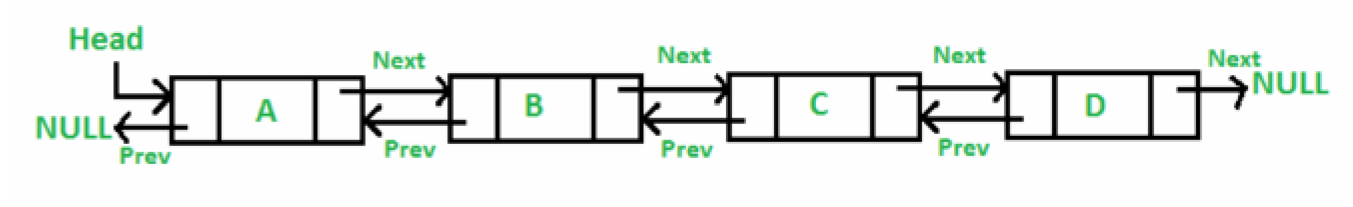
Course: Data Structures (CL2001) **Semester**: Fall 2024
Instructor: Sameer Faisal **T.A**: N/A

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 Linked 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();
};
```

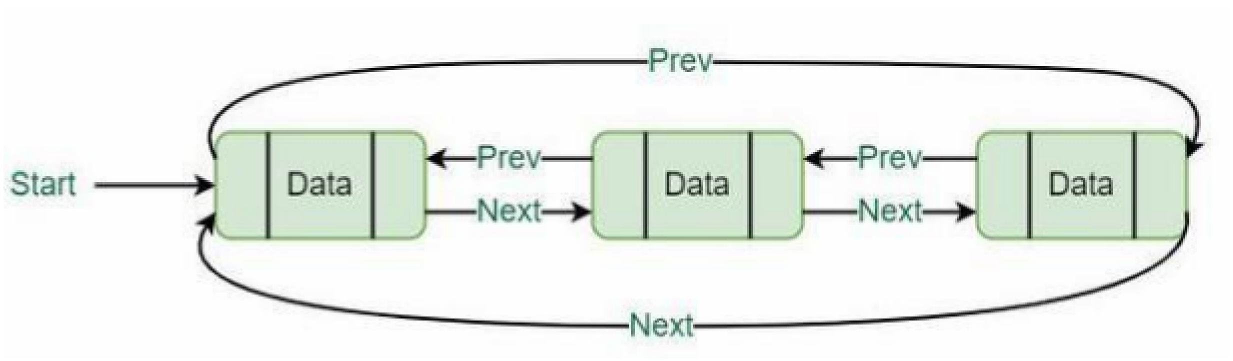
Class Task # 1: Create a doubly link list and perform the mentioned tasks.

(a) Insert a new node at the end of the list.
(b) Insert a new node at the beginning of list.
(c) Insert a new node at given position.
(d) Delete any node.
(e) Print the complete doubly link list.

<u>Circular Doubly Linked List</u>

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;
 }
};
```

Lab Exercises

1. 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.

2. 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.**

3. Given a Double Circular Linked List **1-7-4-2-6-4-5-3-9-8** (You can take your own list as input as well). Your task is to swap the nodes (not values) given their indexes along with its previous and next pointers.
   Input: Enter two nodes keys = 3 7
   Output:
   Initial Linked List = 1-7-4-2-6-4-5-3-9-8
   After Swapping = 1-7-4-3-6-4-5-2-9-8
4. Given a linked list (input by user), you have to perform the following task:

i. Extract the alternative nodes starting from second node.
ii. Reverse the extracted list.
iii. Append the extracted list at the end of the original list.

Note: Try to solve the problem without using any extra memory.

Example:
Input:
LinkedList = 10->4->9->1->3->5->9->4
Output:
10 9 3 9 4 5 1 4

Explanation:
Alternative nodes in the given linked list are 4,1,5,4. Reversing the alternative nodes from the given list, and then appending them to the end of the list results in a list 10->9->3->9->4->5->1->4.