Exp-05: Doubly, circular linked list and their different operations.

a) Objectives:

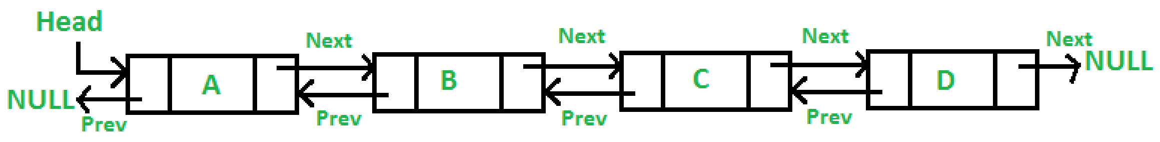
- Doubly Linked List Construction, Traversal
- Doubly Linked List Insertion/Deletion
- Insert value in sorted way in a sorted Doubly Linked List
- Circular Linked List Introduction and Application
- Find out if a linked list is circular or not

b) Prerequisites:

- Pointers
- Structure/Class
- Recursion

c) Theory:

Doubly Linked list: Doubly Linked List is a variation of Linked list in which navigation is possible in both ways, either forward and backward.



- Each node is divided into three fields:
  i) Data filed: contains information of the element
  ii) Next pointer field: contains the address of next node in the list
  iii) Prev pointer field: contains the address of the previous node in the list

- Representation:

```
struct Node {
  int  data;
  Node*  next;
  Node*  prev;
};
```

Task1:

Description:	Construct a doubly linked list by taking inputs from user. Stop adding nodes when user enters -1. Then print the entire list. (Note: we will consider this sample input for the later tasks).
Sample Input:	6 3 2 -1
Sample Output:	6 3 2

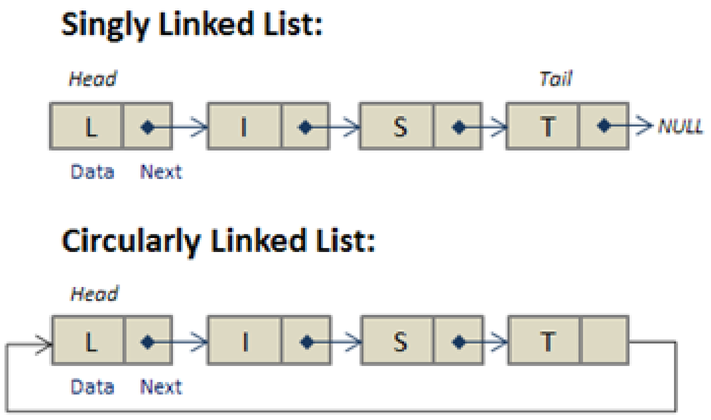
Task2:

Description:	Search for an element in the constructed doubly linked list.	
Sample Input:	3	4
Sample Output:	Found	Not found

Task3:

Description:	Insert an element in a sorted doubly linked list.
Sample Input:	List: 2 5 8 8 10 Add: 7
Sample Output:	List: 2 5 7 8 8 10

Circular linked list: A linked list whose last node points back to the first node instead of containing the null pointer, called a circular linked list.



Applications:

- Full list can be traversed starting from any node.
- Useful for implementing queue. Using circular linked list, we do not need to maintain head, tail separately. We can keep track of tail and the head will always be the next of tail.
- Useful when we want to go repeatedly around the list. For example, when multiple applications are running on the PC, it is common for OS to keep the applications in a list and cycle through them.

Task4:

Description:	Construct a circular linked list by taking inputs from user. Stop adding nodes when user enters -1. Then print the entire list. (Note: we will consider this sample input for the later tasks).
Sample Input:	6 3 2 -1
Sample Output:	6 3 2

Task5:

Description:	Find out if a linked list is circular or not.
Sample Input:	$\begin{tikzcd} 6 \arrow[r] & 3 \arrow[r] & 2 \arrow[r] & 5 \arrow[lll, bend left] \end{tikzcd}$
Sample Output:	Yes

d) Discussion:

- Finding cycle in a linked list. (Floyd's cycle finding algorithm)
- All codes of the lab will be uploaded to google classroom.

e) Homework:

Implement the following tasks on your own. You can discuss with others but copy/pasting code from any source is strictly prohibited. Violation of this rule will result in permanent failure of this course.

1) Perform insertion/deletion operation in doubly linked list.
2) Reverse a doubly linked list.
3) Find the median of a doubly linked list (assume list length is odd).
4) Traverse a circular linked list from any given random node.
5) Implement queue using circular linked list.