Assignment No: 07

Problem Statement:

Appointment Management: Write a menu driven C++ program for storing appointment schedules for the day.
Appointments are booked randomly using linked lists. Set start and end time for visit slots. Write functions for
1. Display free slots
2. Book appointment
3. Cancel appointment (check validity, time bounds, availability etc)
4. Sort list based on time
5. Sort list based on time using pointer manipulation

Objectives:
- To understand use of link list as data structure
- To know the operations on doubly linked lists

Theory :

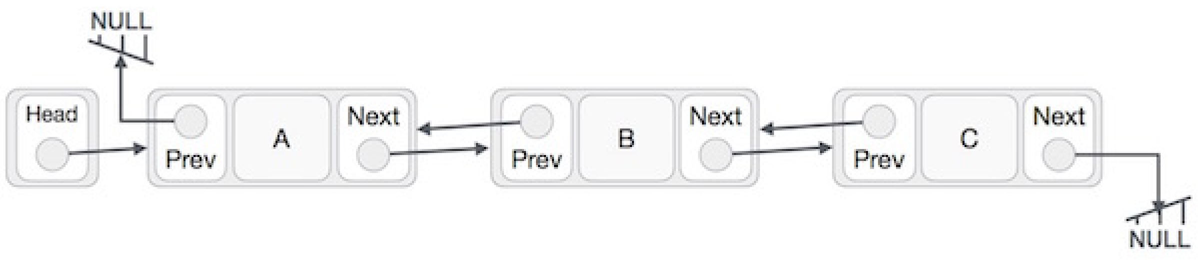
Doubly Linked List:

A doubly linked list is a linked data structure that consists of a set of sequentially linked records called nodes. Each node contains two fields ,called links, that are references to the previous and to the next node in the sequence of nodes. The beginning and ending nodes' previous and next links, respectively, point to some kind of terminator, typically a sentinel node or null, to facilitate traversal of the list.

Doubly Linked List Representation:

Doubly Linked List is a variation of Linked list in which navigation is possible in both ways, either forward and backward easily as compared to Single Linked List. Following are the important terms to understand the concept of doubly 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.
- **LinkedList** − A Linked List contains the connection link to the first link called First and to the last link called Last.



Types of Doubly Linked List:

Following are the various types of linked list.

- **Doubly Linked List** − Items can be navigated forward and backward.
- **Doubly Circular Linked List** − Last item contains link of the first element as next and the first element has a link to the last element as previous.

Basic Operations:

Following are the basic operations supported by a list.

- **Insertion** − Adds an element at the beginning of the list.
- **Deletion** − Deletes an element at the beginning of the list.
- **Insert Last** − Adds an element at the end of the list.
- **Delete Last** − Deletes an element from the end of the list.
- **Insert After** − Adds an element after an item of the list.
- **Delete** − Deletes an element from the list using the key.
- **Display forward** − Displays the complete list in a forward manner.
- **Display backward** − Displays the complete list in a backward manner.

Advantages:

- A DLL can be traversed in both forward and backward direction.
- The delete operation in DLL is more efficient if pointer to the node to be deleted is given.
  In DLL, to delete a node, pointer to the previous node is needed. To get this previous node, sometimes the list is traversed. In DLL, we can get the previous node using previous pointer.

Disadvantages:

- Every node of DLL Require extra space for a previous pointer.
- All operations require an extra pointer previous to be maintained. For example, in insertion, we need to modify previous pointers together with next pointers.

Complexity:

Discuss the time and space complexity of all the functions you have implemented.

Practice Problem:

Write a c++ code for implementing doubly circular link list