Lab 07
Data Structures
BS DS Fall 2024 Morning/Afternoon

Instructions:

- **Attempt the following tasks exactly in the given order.**
- Make sure that there are no ***dangling pointers*** or ***memory leaks*** in your programs.
- Indent your code properly.
- Use meaningful variable and function names. Follow the naming conventions.
- Use meaningful prompt lines/labels for all input/output that is done by your programs.

Task # 0 *(Pre-Requisite)* *(Max Time: 25 Minutes)*

Implement a class for **Circular Doubly Linked List** which stores integers. Your class definitions should look like as shown below:

```
DNode {

      int data;
      DNode* next;
      DNode* prev;
};
```

```
class CDLinkedList {
private:
     DNode *head;

 public:
      CDLinkedList();          // Default constructor
      ~CDLinkedList();         // Destructor
   // Insertion functions
    void insertAtHead(T val);
    void insertAtTail(T val);

    // Deletion functions
    void removeAtHead();
    void removeAtTail();
    void remove(T val);

    // Utility functions
    bool search(T key);
    void update(T key, T val);
    int countNodes();
};
```

Task # 1 *(Max Time: 30 Minutes)*

Implement the following public member function of the **`CDLinkedList`** class:

`void merge (CDLinkedList& list1, CDLinkedList& list2)`

This function will merge the nodes of the two sorted circular doubly linked lists with dummy header nodes (**`list1`** and **`list2`**) to form one sorted list.

For example, if **`list1`** contains **{4 7 10 12}** and **`list2`** contains **{1 3 6 8 9 15}** and **`list3`** is empty, then after the function call **`list3.merge(list1,list2)`**, **`list3`** should contain **{1 3 4 6 7 8 9 10 12 15}** and **`list1`** and **`list2`** should be empty now.

Note: *You are NOT allowed to create any new node in this function. You are also NOT allowed to modify the "data" field of any node. You can assume that the CDLinkedList object on which this function is called is empty at the start of this function.*

Make sure that all boundary cases are properly handled. Also write a drive main function to test the working of your function.

Task # 2 *(Max Time: 25 Minutes)*

Implement the following public member function of the **`CDLinkedList`** class:

`void splitList (CDLinkedList& leftHalf, CDLinkedList& rightHalf)`

This function will split the list (on which it is called) into two halves, and stores these halves in the two lists passed into this function.

For example, if **`list1`** contains **{1 3 5 6 8 12 14}** and **`list2`** and **`list3`** are empty, then after the function call **`list1.splitList(list2,list3)`**, **`list2`** should contain **{1 3 5 6}** and **`list3`** should contain **{8 12 14}** and **`list1`** should be empty now. *Note that if the number of elements are odd, then the one extra element should go into the left half.*

Note: *You are NOT allowed to create any new node in this function. You are also NOT allowed to modify the "data" field of any node. You can assume that the CDLinkedList objects being passed into this function are empty at the start of this function.*

Make sure that all boundary cases are properly handled. Also write a drive main function to test the working of your function.

Task # 3 (Max Time: 30 Minutes)

Implement the **`combine`** and **`shuffleMerge`** functions (see Task # 2 and 3 of Lab # 6) for the **`CDLinkedList`** class. The time complexity of **`combine`** function must be **constant** i.e. $O(1)$.

Also write a drive main function to test the working of your functions.