Skills Assessment 1

Description:
In this skills assessment, you will implement two methods of an existing DoublyLinkedList class namely **isPalindromic** and **swapAtEvenPosition.**

You cannot use any external libraries (such as C++ STL) to develop your code.

starter.zip (Starter Code):
The following is a brief description of the starter code provided to you:

- **ques.cpp**
  - You have been provided with a Node class and a DoublyLinkedList class.
    - The **Node class** contains the following public member variables:
      - **T data:** The data of the node (of templated type)
      - **Node<T>* prev:** The pointer to the previous node
      - **Node<T>* next:** The pointer to the next node
    - The **DoublyLinkedList class** contains the following private member variables
      - **Node<T>* head:** keeps track of the first node of the doubly linked list
      - **Node<T>* tail:** keeps track of the last node of the doubly linked list
      - **int size:** keeps track of the length of the doubly linked list
    - The **DoublyLinkedList class** also implements other methods (**do not change the implementation of these functions**) such as:
      - **void append(T data):** Adds the data at the end of the doubly linked list
      - **string display():** Returns a string that prints out the current state of the doubly linked list (elements from head to tail)

Task
You must implement the **isPalindromic (27 points)** and **swapAtEvenPosition (23 points)** methods in the DoublyLinkedList class.

- The ***bool isPalindromic()*** method should return **true** if the DoublyLinkedList is a palindrome and **false** otherwise.
  - A palindrome refers to a sequence of nodes where the elements, when traversed both forward and backward through the links, form the same sequence. Specifically, the data stored in each node of the doubly linked list remains unchanged when read in both directions. See examples on the next page.
    - Note that **any** doubly linked list with **one node or zero nodes** would satisfy the requirements for being a

palindrome

- **Example:**

```
DoublyLinkedList A: 1 <-> 2 <-> 3 <-> 2 <-> 1
A.isPalindromic()
Output: True

DoublyLinkedList B: 1 <-> 2 <-> 2 <-> 1
B.isPalindromic()
Output: True

DoublyLinkedList C: 1 <-> 2 <-> 3 <-> 2 <-> 2
C.isPalindromic()
Output: False
```

- The ***void swapAtEvenPosition()*** method must swap the data of every node at an even position with the data of its preceding odd positioned node
  - **The position of a node is within the range [1,x] where x is the number of nodes in the doubly linked list**
    - **Therefore, the first node in the list is at position 1 and last node is at position x**
    - **The head points to the node at position 1 and the tail points to the node at position x**
  - For example, you must swap
    - node 2's data with node 1's data
    - node 4's data with node 3's data (and so on)
  - If the doubly linked list has odd number of nodes, then the last node will be left as is (as it does not have a succeeding even positioned node to swap with)

- **Example:**

```
DoublyLinkedList A: 5 -> 4 -> 3 -> 2 -> 6 -> 7
A.swapAtEvenPosition()
A: 4 -> 5 -> 2 -> 3 -> 7 -> 6

DoublyLinkedList B: 5 -> 4 -> 3 -> 2 -> 6
B.swapAtEvenPosition()
B: 4 -> 5 -> 2 -> 3 -> 6
```

You have been provided with a few sample test cases in ques.cpp that test out your implementation of the isPalindromic and

swapAtEvenPosition methods. You can compile your code using the makefile provided and run the executable (./ques).

Deliverables and Submissions:

In a zip folder named using the format **<FirstName>-<LastName>-<UIN>-<SA1>.zip** you must submit the following file to Canvas:

- **ques.cpp (**containing your functional implementation of the isPalindromic and swapAtEvenPosition methods)

Do not use angular brackets in the name of the submission folder.