Lab 5: Doubly-linked List

Using the **start project**, implement the functions listed below. Write the **declaration** in the **DoublyList.h** file and the **implementation** in the **Functions.cpp** file, where indicated.

- **DoublyList** member function **print**
  - Prints all the elements in the list, starting from the first element.
  - Assume the calling object is a non-empty list.

- **DoublyList** member function **reversePrint**
  - Prints all the elements in the list in reverse, starting from the last element.
  - Assume the calling object is a non-empty list.

- **DoublyList** member function **front**
  - The function returns the data stored in the first node.
  - Assume the calling object is a non-empty list.

- **DoublyList** member function **back**
  - The function returns the data stored in the last node.
  - Assume the calling object is a non-empty list.

- **DoublyList** member function **copyToList**
  - **Parameter:** an object of the class **DoublyList**
  - Copies all the nodes from the calling object into the parameter object.
  - Use the function **insertFront** of the same class.
    - **NOTE:** We are calling the function **insertFront**, which is a **member** of the **same** class, to avoid re-writing several lines of code. If **insertFront** had only a couple of statements, then it would have been more efficient to re-write the statements instead of calling **insertFront**.
  - **Assumptions:**
    - The calling object is non-empty.

- Member function **insertInOrder**
  - **Parameter:** an integer to insert
  - Inserts the item in ascending order. The calling list is either empty or, if it contains elements, they are already in order, which means that you need to find the correct position where to insert.
  - For example:
    - The list has no nodes => The element to insert will be stored in a node that will become the first and only node in the list.
    - The list has at least one node => The element to insert will be stored in a node that needs to be positioned in the correct order: Element is 5, list is 2 4 7 8, then the list will become 2 4 5 7 8.

The **main** function in the **Main.cpp** file contains a few testing cases. You will need to create testing cases for the **copyToList** function. Make sure you try different scenarios:

1. The calling object has more elements than the parameter object.
2. The calling object has less elements than the parameter object
3. The calling object and the parameter object have the same number of elements.

EXPECTED OUTPUT: