Practice Exercise 4: Doubly-linked List

Using the start project for practice exercise 4, implement the following functions:

- Member function **print**
  - Prints all the elements in the list, starting from the first element.
  - Consider the case when the list is empty and output the error message, "List is empty."

- Member function **reversePrint**
  - Prints all the elements in the list in reverse, starting from the last element.
  - Consider the case when the list is empty and output the error message, "List is empty."

- Member function **front**
  - The function returns the data stored in the first node.
  - Consider the case when the list is empty and output the error message, "List is empty. Cannot access front node." If the list is empty, you do not have an integer to return; therefore, gracefully exit the program by using the statement **exit(1).**

- Member function **back**
  - The function returns the data stored in the last node.
  - Consider the case when the list is empty and output the error message, "List is empty. Cannot access last node." If the list is empty, you do not have an integer to return; therefore, gracefully exit the program by using the statement **exit(1).**

- 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 a function within the same class to avoid re-writing several lines of code. We would not call a function that has only a couple of statement, rather we would re-write the statements (for example, a function that returns true if the list is empty should not be called from the same class, because it is easier to write if (first == NULL)).

- Member function **insertInOrder**
  - **Parameter:** an integer to insert
  - Inserts the item in ascending order.

Create your testing cases. Make sure you consider all possibilities (empty list, list with one node, list with two nodes, list with several nodes).