Lab 8: The Big Three

For this lab, you need to create a new project and import the given files—**DoublyLIst.h**, **DoublyList.cpp**, **Main.cpp**, and **Functions.cpp**—into the project. The **Main.cpp** file contains testing cases; the **DoubyList.h** file contains the class **Node** completed and the class **DoublyList** partially completed; the **DoublyList.cpp file contains** the implementation of some of the functions needed for testing. The class **DoublyList** creates doubly-linked lists by inserting nodes to the back of the list. The class **Node** includes a function that **overloads the comparison operator**. This is needed to compare two nodes.

Your job is to implement all the functions listed below. You will need to write all the **declarations** in the **DoublyList.h** file and all the **definitions** in the **Functions.cpp** file. (Note that **NO** parameters or return values/references are specified, which means that you need to figure out from the function call how to declare your functions.)

- **Overloaded insertion operator**
  - This function prints the list on the same line with each element separated by a space.
  - If the list is empty, the function prints the error message, "List is empty."

- **Copy constructor**
  - Creates an object that is an exact copy of the parameter passed.

- **Overloaded assignment operator**
  - Include an **IF/ELSE** statement that checks whether the calling object and the parameter are the same list. If they are, output the error message, "Attempted assignment to itself." If the two objects are different, then you need to copy the elements in an **efficient** way by considering the following cases:

    - The calling list is empty.
      - Use the **insertBack** function to start inserting nodes in the calling lists that store the same data that the parameter lists is storing.

    - The calling object has more nodes than the parameter object.
      - Copy the data from the parameter object into the calling object, and then delete the additional nodes left in the calling object.

    - The calling object and the parameter object have the same number of nodes.
      - Since all the nodes are already there, you simply need to copy the data from the calling object into the parameter object.

    - The calling object has fewer nodes than the parameter object.
      - Copy the data from the parameter object into the calling object, and then create additional nodes (use the **insertBack** function) to copy the rest of the parameter list.

    - The parameter object is empty.

- Use the function **destroyList** to delete all the nodes in the calling object.

Once you have completed the implementation, you will need to **answer the 3 questions** at the top of the **Functions.cpp** file.

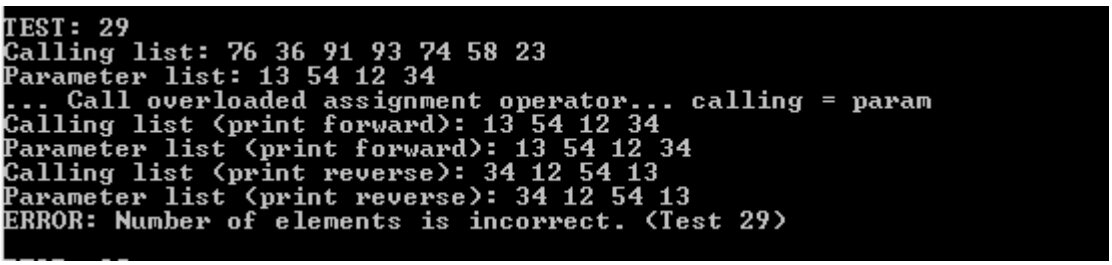
Do **NOT** remove any of the comments.

IMPORTANT

- The purpose of this assignment is for you to understand the **difference** between a **copy constructor** and an **assignment operator**, and to further understand how to use the **debugger**.

If there are any errors in your implementation, the **testing** will detect them:

- Some errors might crash your programs
- Other errors might be caught by the testing file as shown below.



The **ERROR** label identifies the bug originating in Test 29, showing that the number of elements is not correct. The output.exe file shows the expected output.