THE UNIVERSITY OF NEWCASTLE AUSTRALIA

Discipline of Computing and Information Technology
Semester 2, 2021 - SENG1120/6120

Assignment 1

Due using the Canvas Assignment submission facility:
11:59PM – Sunday September 4th, 2022

version 1.0.1

NOTE: *There is important information about submission and code specifics at the end of this assignment specification***.**

INTRODUCTION

You are required to build the infrastructure to manipulate data related to student scores. Your client further specifies that you are to create a class named `LinkedList` to store the students' information. The `LinkedList` will store each name of the student and their score in a `Node` of the list, using the provided class `Student`.

ASSIGNMENT TASK

You are required to create your own implementation for the `LinkedList` class (*including all functions* – not just those required by the demo file) as a doubly-linked list, as discussed in lectures. It will use instances of `Node` to store instances of `value_type` (*in this assignment, each* `Node` *will be used to store an instance of* `Student`).

The `LinkedList` class will be used by a main program, to be supplied to you, as well as a `makefile`.

You will need to design `LinkedList` and `Node` in a way that it communicates seamlessly with the main program and the class `Student` provided, and compiles with the `makefile` also supplied. Please refer to the lecture slides and recordings for guidance on how to implement both the `LinkedList` and `Node` classes.

For students in SENG6120, there is an extra requirement:

- (3.0 marks) Implement the member function `void order()` inside `LinkedList`. That method will order the names of the students in alphabetical order.

- You are **NOT ALLOWED** to manipulate the contents of the `Node`'s `value_type` variable. You can only manipulate the *pointers of the nodes* to move them around until the list is ordered.
- In addition, you are **NOT ALLOWED** to instantiate new nodes in the implementation of the function `void order()`.
- Finally, you are **REQUIRED** to overload the `operator <` for `Student`, and use it in the `order()` function. As this will be a non-member function, you may implement this within the `LinkedList.h/.cpp` files (*but outside the* `LinkedList` *class definition*)

For SENG1120 students who want to be challenged more, the above requirement becomes a bonus question, also worth 3.0 marks; however you can still only score a MAXIMUM of 15.0/15.0.

SUBMISSION

Make sure your code works with the files supplied, and DO NOT change them. For marking, we will add the main file and the `Train` class to the project and compile everything using the `makefile`, together with your own files. **If it does not compile or run, your mark will be zero (as we are unable to test and validate your implementation).**

Your submission should be made using the Assignments section of the course Blackboard site. **Incorrectly submitted assignments will not be marked.** You should provide the `.h` and `.cpp` files related to the `LinkedList` and `Node` classes only. Also, if necessary, provide a `readme.txt` file containing instructions or comments for the marker. Each program file should have a proper header comment section including your name, course and student number; and your code should be properly documented.

Remember that your code should compile and run correctly using gcc and Cygwin. There should be no segmentation faults or memory leaks during or after the execution of the program.

Compress all your files into a *single .zip file*, using your **student number** as the filename. For example, if your student number is **`c9876543`**, you would name your submission:

`c9876543.zip`

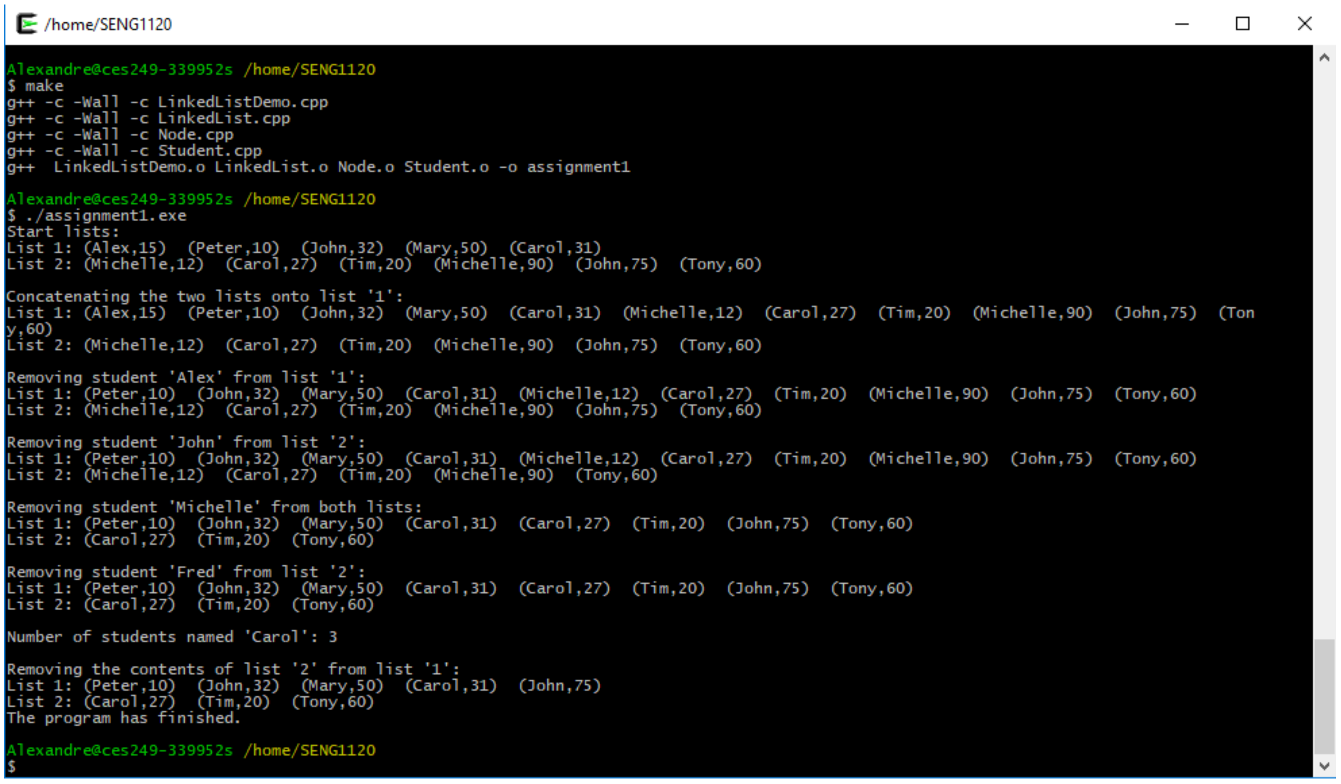
If you have attempted the Bonus Requirement (*or you are a 6120 student*), please include a blank text file in the same folder as your source files, simply called **`Bonus.txt`** – this is to make it clear to the marker that you are attempting this.

Submit by selecting the **Assignment 1** link that will be found in the **Assessment** section on **Canvas**.

Late submissions are subject to the rules specified in the Course Outline. Finally, a completed **Assignment Cover Sheet** should accompany your submission.

This assignment is worth 15 marks of your final result for the course (*including bonus marks*).

Compiling and running your files together with the demo file provided should output the following result:



```
Alexandre@ces249-339952s /home/SENG1120
$ make
g++ -c -Wall -c LinkedListDemo.cpp
g++ -c -Wall -c LinkedList.cpp
g++ -c -Wall -c Node.cpp
g++ -c -Wall -c Student.cpp
g++  LinkedListDemo.o LinkedList.o Node.o Student.o -o assignment1

Alexandre@ces249-339952s /home/SENG1120
$ ./assignment1.exe
Start lists:
List 1: (Alex,15)  (Peter,10)  (John,32)  (Mary,50)  (Carol,31)
List 2: (Michelle,12)  (Carol,27)  (Tim,20)  (Michelle,90)  (John,75)  (Tony,60)

Concatenating the two lists onto list '1':
List 1: (Alex,15)  (Peter,10)  (John,32)  (Mary,50)  (Carol,31)  (Michelle,12)  (Carol,27)  (Tim,20)  (Michelle,90)  (John,75)  (Ton
y,60)
List 2: (Michelle,12)  (Carol,27)  (Tim,20)  (Michelle,90)  (John,75)  (Tony,60)

Removing student 'Alex' from list '1':
List 1: (Peter,10)  (John,32)  (Mary,50)  (Carol,31)  (Michelle,12)  (Carol,27)  (Tim,20)  (Michelle,90)  (John,75)  (Tony,60)
List 2: (Michelle,12)  (Carol,27)  (Tim,20)  (Michelle,90)  (John,75)  (Tony,60)

Removing student 'John' from list '2':
List 1: (Peter,10)  (John,32)  (Mary,50)  (Carol,31)  (Michelle,12)  (Carol,27)  (Tim,20)  (Michelle,90)  (John,75)  (Tony,60)
List 2: (Michelle,12)  (Carol,27)  (Tim,20)  (Michelle,90)  (Tony,60)

Removing student 'Michelle' from both lists:
List 1: (Peter,10)  (John,32)  (Mary,50)  (Carol,31)  (Carol,27)  (Tim,20)  (John,75)  (Tony,60)
List 2: (Carol,27)  (Tim,20)  (Tony,60)

Removing student 'Fred' from list '2':
List 1: (Peter,10)  (John,32)  (Mary,50)  (Carol,31)  (Carol,27)  (Tim,20)  (John,75)  (Tony,60)
List 2: (Carol,27)  (Tim,20)  (Tony,60)

Number of students named 'Carol': 3

Removing the contents of list '2' from list '1':
List 1: (Peter,10)  (John,32)  (Mary,50)  (Carol,31)  (John,75)
List 2: (Carol,27)  (Tim,20)  (Tony,60)
The program has finished.

Alexandre@ces249-339952s /home/SENG1120
$
```

Note: the output above is missing the Averages, shown in the text sample output on the next page!

```
Alexandre@ces249-339952s /home/SENG1120
$ ./assignment1.exe
Start lists:
List 1: (Alex,15)  (Peter,10)  (John,32)  (Mary,50)  (Carol,31)
List 2: (Michelle,12)  (Carol,27)  (Tim,20)  (Michelle,90)  (John,75)  (Tony,60)

Concatenating the two lists onto list '1':
List 1: (Alex,15)  (Peter,10)  (John,32)  (Mary,50)  (Carol,31)  (Michelle,12)  (Carol,27)
(Tim,20)  (Michelle,90)  (John,75)  (Tony,60)
List 2: (Michelle,12)  (Carol,27)  (Tim,20)  (Michelle,90)  (John,75)  (Tony,60)

Removing student 'Alex' from list '1':
List 1: (Peter,10)  (John,32)  (Mary,50)  (Carol,31)  (Michelle,12)  (Carol,27)  (Tim,20)
(Michelle,90)  (John,75)  (Tony,60)
List 2: (Michelle,12)  (Carol,27)  (Tim,20)  (Michelle,90)  (John,75)  (Tony,60)

Removing student 'John' from list '2':
List 1: (Peter,10)  (John,32)  (Mary,50)  (Carol,31)  (Michelle,12)  (Carol,27)  (Tim,20)
(Michelle,90)  (John,75)  (Tony,60)
List 2: (Michelle,12)  (Carol,27)  (Tim,20)  (Michelle,90)  (Tony,60)

Removing student 'Michelle' from both lists:
List 1: (Peter,10)  (John,32)  (Mary,50)  (Carol,31)  (Carol,27)  (Tim,20)  (John,75)
(Tony,60)
List 2: (Carol,27)  (Tim,20)  (Tony,60)

Removing student 'Fred' from list '2':
List 1: (Peter,10)  (John,32)  (Mary,50)  (Carol,31)  (Carol,27)  (Tim,20)  (John,75)
(Tony,60)
List 2: (Carol,27)  (Tim,20)  (Tony,60)

Average of list '1': 38.125
Average of list '2': 35.667

Number of students named 'Carol': 3

Removing the contents of list '2' from list '1':
List 1: (Peter,10)  (John,32)  (Mary,50)  (Carol,31)  (John,75)
List 2: (Carol,27)  (Tim,20)  (Tony,60)
The program has finished.

Alexandre@ces249-339952s /home/SENG1120
$
```