University of Newcastle
Discipline of Computing and Information Technology
Semester 2, 2019 - SENG1120/6120

Assignment 1

Due using the Blackboard Assignment submission facility:
11:59PM – September 6th, 2019

NOTE: *The 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 use a **doubly-linked list**, as discussed in lectures, to create your own implementation of the `LinkedList` class. 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 classes.

For students in SENG6120, there is an extra requirement:

- (3.0 marks) Implement the member method `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 method `void order()`. Finally, you are REQUIRED to overload the < operator for `Student`, and use it in the `order()` method.

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 Student 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.

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 linked list and node classes, only, plus an assessment item coversheet. Also, if necessary, provide a `readme.txt` file containing instructions for the marker. Each program file should have a proper header 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 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, and appending '**_order**' if you have attempted the bonus section. For example, if your student number is **c9876543** and you have implemented `order()` (*including 6120 students*), you would name your submission:

`c9876543_order.zip`

The same student who has NOT attempted the order() task would simply name their submission:

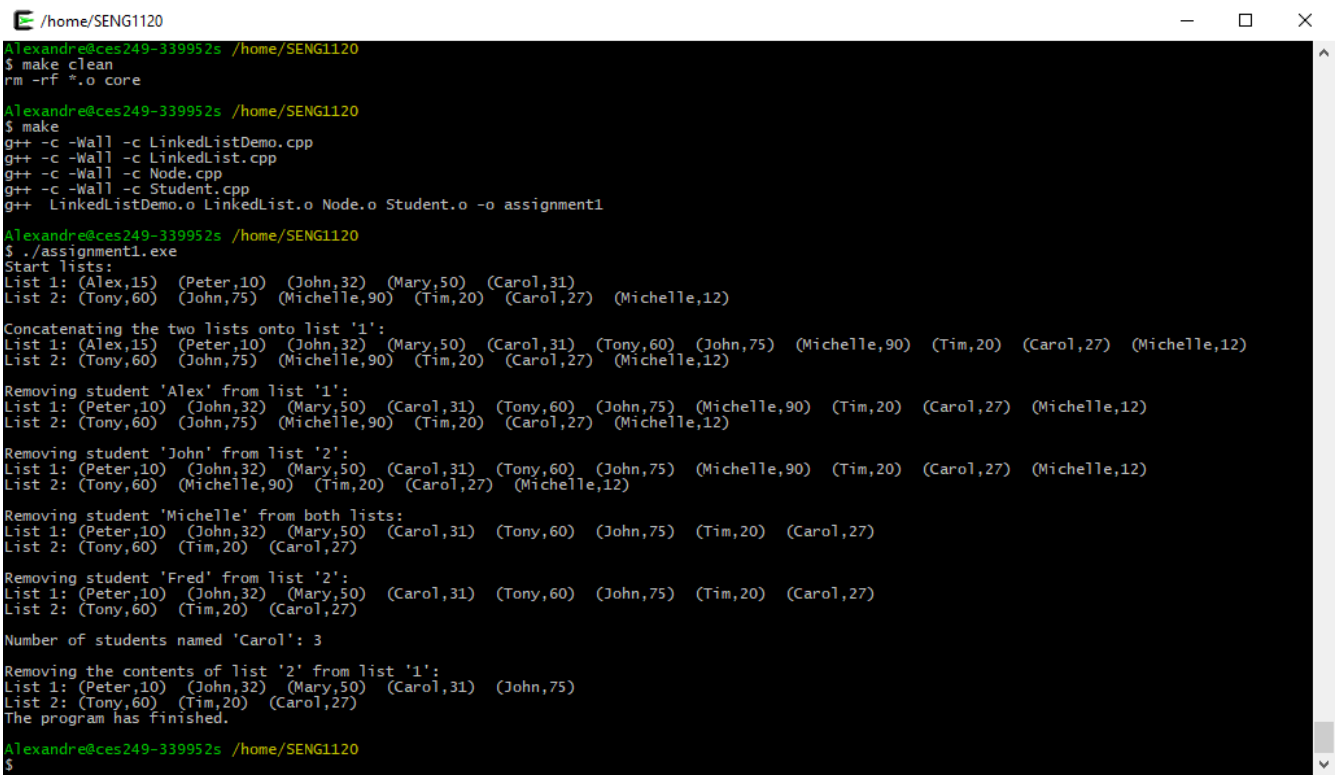
`c9876543.zip`

Submit by clicking in the link that will be created in the Assignments section on Blackboard.

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.

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



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

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

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

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

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

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

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: (Tony,60)  (Tim,20)  (Carol,27)
The program has finished.

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

Dan
v1.0 (*2019-08-14*)
v1.01 (*2019-08-21*) – clarified the use of a **doubly-linked list** on page 1.