University of Newcastle
Discipline of Computer Science and Software Engineering
Semester 1, 2020 - SENG1120/6120

Assignment 2

Due using the Blackboard Assignment submission facility:
11:59PM – Sunday, May 17th, 2020

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

INTRODUCTION

In lectures we have discussed the use of `templates` to provide the compiler with blueprints for functions and classes. These are used by the compiler to produce code that implements functions and/or classes that are suitable for the type(s) to which you apply them in your program code.

PROBLEM DESCRIPTION

Your task in this Assignment is to implement a traditional board game - the ***Tower of Hanoi***. The Tower of Hanoi is a mathematical puzzle consisting of three rods, and several disks of different sizes that can slide onto any rod. The puzzle starts with the disks neatly stacked in order of size on the left rod, the smallest at the top, thus making a conical shape.

The objective of the puzzle is to move the entire stack to the right rod, obeying the following rules:

- Only one disk may be moved at a time.
- Each move consists of taking the upper disk from one of the rods and sliding it onto another rod, on top of the other disks that may already be present on that rod.
- No disk may be placed on top of a smaller disk.

For complete information about it, and to play it, visit the following websites:
https://www.mathsisfun.com/games/towerofhanoi.html
http://en.wikipedia.org/wiki/Tower_of_Hanoi

ASSIGNMENT TASK

In this assignment, you will produce a:

1. class `DiscInt`, which will be used to represent the discs as integers and have an `int` as its member variable.
2. class `DiscString`, which will be used to represent the discs as strings and have a `string` as its member variable
3. class template `Node`, which will store an instance of `DiscInt/DiscString` and is then used as a component in the construction of

4. class template `LinkedList`, which is a Doubly-Linked List and is then used as a component in the construction of
5. class template `LStack`, which is then used as a component in the construction of
6. class template `TowerHanoi`, which implements the functionality of the game.

`Node` and `LinkedList` should be dynamic (use *pointers*).

You will use the `TowerHanoiDemo` file provided, which will interface with the user to get the initial configuration and then start the game. We are also providing a `makefile`.

Having created and confirmed the correct operation of class templates, you will implement the Tower of Hanoi as follows. `TowerHanoi` will create three instances of `LStack`. Each will correspond to one rod.

Your implementation will also allow the game to perform a check and only move a disc if the movement is allowed. If the movement is not allowed, you should display a warning message, in the form "Invalid Move, Try Again!".

To allow the game to be displayed (and played), the classes `DiscInt`/`DiscString` should be able to be initialized with a certain value (1/"X", 2/"XXX", 3/"XXXXX" and so on), print themselves (<<) and make comparisons (<).

Next, we show examples of how the game would look like for the two types of discs:

Tower of Hanoi with `DiscString`: If the `Nodes` will be displayed as strings, printing the game state with 5 discs should produce the same output as below:

```
      X
     XXX
    XXXXX
   XXXXXXX
  XXXXXXXXX
---------------------------------
      1         2         3
```

Suppose the player moves a disc from rod 1 to rod 3. When the game is printed the next time, the output will be:

```
     XXX
    XXXXX
   XXXXXXX
  XXXXXXXXX                X
---------------------------------
      1         2         3
```

The game is considered complete when the discs have been successfully moved to the third rod, as:

```
                              X
                             XXX
                            XXXXX
                           XXXXXXX
                          XXXXXXXXX
---------------------------------
       1          2          3
```

Tower of Hanoi with `DiscInt`: If the `Nodes` will be displayed as integers, printing the game state with 5 discs should produce the same output as below:

```
       1
       2
       3
       4
       5
---------------------------------
       1          2          3
```

Suppose the player moves a disc from rod 1 to rod 3. When the game is printed the next time, the output will be:

```
       2
       3
       4
       5                     1
---------------------------------
       1          2          3
```

The game is considered complete when the discs have been successfully moved to the third rod, as:

```
                              1
                              2
                              3
                              4
                              5
---------------------------------
       1          2          3
```

Extended question for SENG6120 or BONUS question for SENG1120 (2 bonus marks)

Instead of having the three rods stored in an array, or as separate variables, the implementation of TowerHanoi must use the C++ container `List<LStack>`.

SUBMISSION

Make sure your code works with the files supplied, and DO NOT change them. For marking, we will add the `TowerHanoi` file 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`, `.cpp` and `.hpp` files related to the `TowerHanoi`, `LStack`, `LinkedList`, `Node`, `DiscInt` and `DiscString` 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 '**_bonus**' if you have attempted the bonus section. For example, if your student number is **c9876543** and you have implemented the **Extended Question** (*including 6120 students*), you would name your submission:

`c9876543_bonus.zip`

The same student who has NOT attempted the **Extended Question** 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 10 marks of your final result for the course.

Alex and Dan – v1.0 (2020-04-09)