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 template `Node`, which is then used as a component in the construction of
2. class template `LinkedList`, which is then used as a component in the construction of
3. class template `LStack`, which is then used as a component in the construction of
4. `TowerHanoi`, which implements the functionality of the game. This is a normal class, not a template one.

`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. Also, the discs will be implemented as strings. That is, `Nodes` will store strings representing the discs:

```
“    X    “
“   XXX   “
“  XXXXX  “
“ XXXXXXX “
“XXXXXXXXX“
```

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
```

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` 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.