COSC 220 - Computer Science II
Project 2 - Towers of Hanoi

Dr. Joe Anderson

Due: 3 November 2019

1 Description

The "Towers of Hanoi" is a mathematical puzzle where a user is presented with a set of tori (or, disks with holes in the center) of all different diameters, and three poles. The tori (disks) are initially stacked in pyramid-fashion (largest diameter) on the first pole. The task of the user is to move the disks, one at a time, never placing a larger disk on a smaller one, until the pyramid has been reconstructed on one of the other poles.

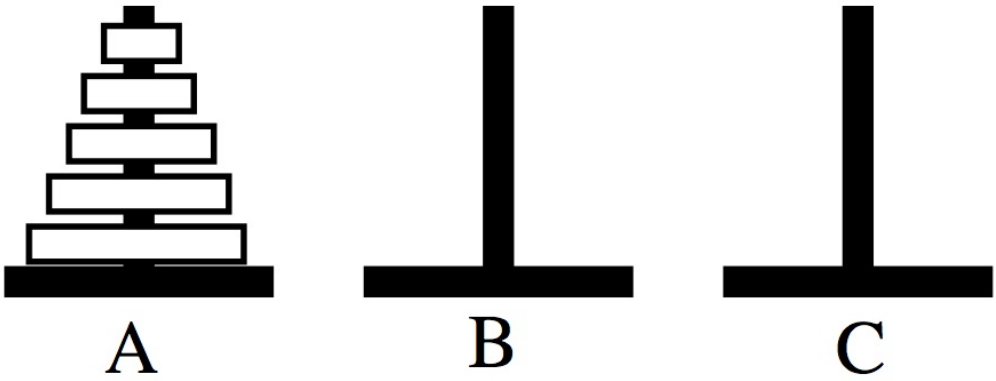
The rules of the game are:

- You can only move one disk at a time (i.e. you may only take the top disk from a single stack, and must place it onto a stack before taking another disk).
- No disk can be placed over a smaller one.

2 Specifications

1. Write a program that allows the user to change how many disks are initially in the puzzle (within a reasonable limit, say, maximum of 10).
2. Your program should have a class called `HanoiStack` which implements the stack functionality of pushing and popping disks.
    - The only public members of `HanoiStack` should be the constructor, a function to push (taking a `Disk` object or pointer as argument), a function to pop (returning either `Disk` or returning by reference), and a function to print or display the current configuration of the tower.

Figure 1: Initial setup of the Towers of Hanoi with five disks.



(a) Add an overloaded `operator +=` as another way to a disk.

- If you want to add functionality that you think requires extra publicly accessible attributes, they must be approved by the instructor.
- You may implement any *private* attributes of `HanoiStack` that you think you need.

3. The elements of the stack should be a `struct` called `Disk` which have an integer attribute for the size of the disk.
4. The program should allow the user to choose a stack to take a disk from, and then select where to place the disk.
5. The `HanoiStack` should also enforce the rule that a disk cannot be placed on top of a smaller one.
6. The program structure should use three instances of the `HanoiStack` class, where only one of them has disks at the start, and they are correctly placed with the largest at the bottom.
7. The program should have a way to display to the user the current state of the three stacks after each action (not necessarily graphical. See bonus below.)
   (a) Add an overloaded `operator <<` to make this simpler to use.
8. Add an option for the program to print out the optimal solution (without actually making the moves), by using the recursive algorithm discussed in class.
9. **DO NOT** use the C++ standard library `stack` class.
10. User input should be reasonably validated.

3 Submission

Upload your project files (for this project only!) to the course canvas system.

Turn in (stapled) printouts of your source code, properly commented and formatted with your name, course number, and complete description of the code.

Also turn in printouts reflecting several different runs of your program (you can copy/past from the terminal). Be sure to test different situations, show how the program handles erroneous input and different edge cases.

4 Bonus

You may or may not add the following features for a maximum of 5 extra points each:

- Add a graphical representation of the individual towers.
- Add a "win-condition" to detect that the original tower has been moved, correctly, to a new location.
- (Only if you complete the previous feature) Count the number of moves that the user used to win.