CS 105 (C++)

Assignment 6: The Tower of Hanoi

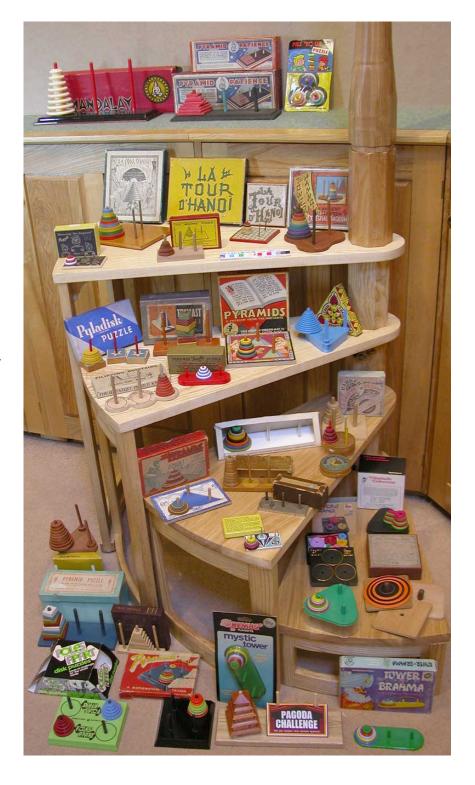
I. Overview

In this assignment, you will complete the implementation of a text-based <u>Tower of Hanoi</u> game by creating a dynamically allocated stack class. (As in Assignment 4, the stack's internal storage will be implemented as a linked list, with the head of the list corresponding to the top of the stack.)

You will use a6_main.cpp (download here) that handles game logic and display, and you will implement your class Stack within the files Stack.h and Stack.cpp. The given a6_main.cpp should not be modified; doing so may cause your code to work incorrectly during grading.

Your implementation must meet the following requirements:

- Stack.h must properly use preprocessor directives to prevent multiple inclusion.
- Stack.h must contain a complete definition of a helper class to be used for nodes in Stack's internal linked list. All member functions in the helper class should be defined when
 - they are declared (i.e., within the class definition).
- Your helper class must contain private data members for an integer value and a next pointer, along with public member functions to get and set each one.
- Stack.h must contain a declaration for class Stack, but all of Stack's member functions must be defined in Stack.cpp.
- The Stack class must contain private data members for current stack height and a pointer to the head of the internal linked list.
- The Stack class must contain the following public member functions:



- A constructor to initialize the private data members. (The list head should be 0 when the list is empty.)
- A destructor to deallocate any memory used for the internal linked list.
- o push, which takes an int and adds it to the top of the stack and returns nothing.
- o pop, which removes and returns the top value from the stack.
- o peek, which returns the value at the top of the stack, but does not remove it.
- isEmpty, which returns a boolean value to indicate whether the stack is empty.
- o getHeight, which returns the current height of the stack as an integer.
- As an exercise in programming to specifications, the Stack class and its helper class should contain exactly the members specified, and no others.
- As with Assignment 4, proper handling of dynamic memory (allocation, deallocation, etc.) will be extremely important because of the potential for damaging and hard-to-find bugs. (As always, I recommend drawing an illustration of dynamic memory operations to ensure correctness.)

If your Stack class is properly implemented, you will be able to compile it with the given a6_main.cpp to produce a working simulation of the Tower of Hanoi game. There's a detailed description at the Wikipedia link above, but basically, your goal is to transfer the tower from the first peg to the last by moving one disc at a time, with the restriction that you can never place a larger disc on top of a smaller one. Be sure to test your completed implementation thoroughly.

You can compile the entire program by listing the .cpp files as arguments to g++ as follows:

```
g++ a6_main.cpp Stack.cpp -o a6
```

II. Grading

The following is a list of specific assignment requirements, along with the grade value for each (out of a total of 10 points for the assignment).

• Minimum Requirements

- Stack functions properly within the game defined in a6 main.cpp.
- Your work must be submitted in files named Stack.h and Stack.cpp.
- Your Stack.h and Stack.cpp (when combined with the given a6_main.cpp) must compile on a department UNIX machine with the following command:

```
g++ a6 main.cpp Stack.cpp -o a6
```

• Before evaluation, your Stack.h and Stack.cpp must be submitted via turnin, using the following command on a department UNIX machine:

```
turnin -- submit dlessin a6 Stack.h Stack.cpp
```

Graded Elements

- Preprocessor directives used to prevent multiple inclusion in Stack.h.
- Helper class defined completely within Stack.h, and with all member functions defined as they are declared (within the class definition).
- Helper class contains all specified members (and no others).
- Stack class divided between Stack.h and Stack.cpp as described in Section I.
- stack class contains all specified members (and no others).

- \circ Proper use of Stack constructor as described in Section I.
- Proper use of Stack destructor as described in Section I.

• Provisional Dealbreakers

- Due to the importance of proper handling of dynamic memory, you will lose all points from the Proper Function portion of your grade if a valgrind evaluation of your work shows any memory leaks or errors.
- **However**, in recognition of the difficulty of this task, you may resubmit your work at any time before the end of the course for regrading on this portion of your score.
- Valgrind evaluation is performed as follows:
 - Prepare your executable for valgrind by compiling with debugging information on and optimization off. Use the following command for this (Note that "00" is a capital letter "o" followed by the number zero.):

```
g++ -g -00 a6 main.cpp Stack.cpp -o a6
```

Run valgrind by prepending the following to your normal command (including all normal arguments):

```
valgrind --leak-check=yes
```



This work is licensed under a <u>Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported</u> License.