University of Wollongong
School of Computer Science and Software Engineering

CSCI124/MCS9124 Applied Programming Autumn 2015

Lab 11 (1 mark)
Due at 11:59pm on Thursday, 28th May.

Background:
Use the generic stack class as part of a simple game management class.

Task:
In the previous week's lab you implemented a simple stack class. In this task you will use the stack class to represent a needle in a Towers of Hanoi game. An implementation of the Stack class is provided in the files `stack.h` and `stack.cpp`, or you can use your own from last week's lab. The data type being stored in the stack should be specified in an interface file `dataType.h`.

First we will extend our class stack slightly to provide a method `print(ostream&)` , which prints each element of the stack (from 0 to top) separated by spaces, and a method `peek()` which returns the top element of the stack.

hanoi.h:
Create a class called Hanoi in the file `hanoi.h`.
The class will use an array of 3 integer stacks to represent the needles A, B and C.

The principal method provided by this class will be `run` which will run a single game. It should take no arguments, and return `true` if the game was won or `false` if the game was quit.

There should also be a `rules` method to print the rules of a towers of hanoi game.

Some auxiliary methods will also be required by `run`. These include:

- a boolean `setup` method to initialize or reset the game state,
- a `print` method to print the current state of the game,
- a boolean method `isSolved` to test whether the end state has been reached,
- a `moveDisc` method, taking values in the range 0-2, which will make a single move `from` a needle `to` another needle, and
- a boolean `move` method, which requests a move from the user and makes the move.

main.cpp:
Create a main program to create a Hanoi object and call its run method.

hanoi.cpp

The implementations for the methods should be placed in `hanoi.cpp`

The `setup` method ensures that all stacks are empty, then requests from the user a number of discs for the game and initializes the stack corresponding to needle A to contain the appropriate discs, then return true. (Note that we couldn't easily have more than 20 discs because our Stack does not automatically resize – although this *could* be worked around with either an array of Stack* or by using placement new). If the user enters `'q'`, (you may want to peek() to check this) the method should return false, otherwise it should continue to reprompt until a valid number of discs is entered.

The print method should print the stack for each needle, preceded by the character identifying it (A-C)
For example, a game in which stack A contains discs 4-2 and stack C contains disc 1:

```
A 4 3 2
B
C 1
```

For now, have `run` just call `setup` and `print`, to check that everything is working.

```
How many discs? 5
A 5 4 3 2 1
B
C
```

The `moveDisc` method should make the specified move if possible and return `true`, or return `false` if the move is not possible. Recall that a larger disc is not permitted to be placed on a smaller disc.

The `move` method repeatedly prints the current state and requests a move from the user until a valid move is made, at which point it returns true. You may want to control your loop using a flag which stores the return value of the `moveDisc` method. If at any point the user types `'q'`, the method should return false.

The `isSolved` method should return `true` if all discs are on the final stack. It should not pop or push anything in order to do this.

Extend `run` so that it continues to make moves until the puzzle is solved. If at any time move returns `false`, run should also return false. Once the puzzle is solved, `run` should return `true`.

Multiple games

Modify the main program to print the rules, ask if the user wants to play again, and repeatedly run the game as long as the user responds with 'y'. A sample output for the program is provided on the next page.

Submission:

You are to submit the files:

```
hanoi.h, hanoi.cpp, stack.h, stack.cpp, dataType.h, main.cpp
```

using the submit program:

```
$ submit –c csci124 –a lab11 –u <username> <files>
```

Sample Output

```
Welcome to the Towers of Hanoi
The aim is to move all discs from tower A to tower C
You may not move a larger disc on top of a smaller disc
You can type q at any time to quit

How many discs? q
Oh well, never mind
play again(y/n)?
y

How many discs? 4
A 4 3 2 1
B
C
Move from q
Oh well, never mind
play again(y/n)? y

How many discs? 2
A 2 1
B
C
Move from a
Move to a
Cannot move from a to a
A 2 1
B
C
Move from A
Move to b
A 2
B 1
C
Move from a
Move to C
A
B 1
C 2
Move from b
Move to c
A
B
C 2 1
Congratulations, you won!
play again(y/n)? n
```