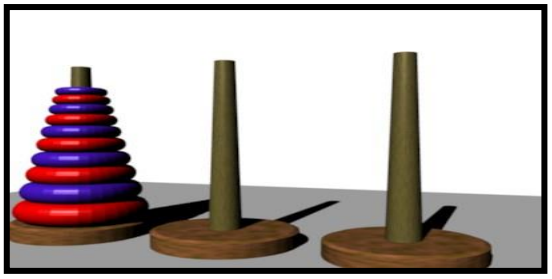
CSCE 2211 Spring 2024 Applied Data Structures
Assignment #1

Dr. Amr Goneid ***Date: Thu Feb 8, Due: Thu Feb 15, 2024***

Implement the following programs using the **Stack ADT**:

Problem 1: The Towers of Hanoi (50 points)



In the Towers of Hanoi game, there are 3 pegs (A , B , C) and N disks with varying sizes that can be stacked on a peg. The objective is to move all the disks from peg (A) to peg (C), probably by using the auxiliary peg (B). At any moment, no larger disk can be placed on top of a smaller one. For example:

- To move one disk from A to C:
  Move disk1 from A to C
- To move two disks (top is 1, bottom is 2):
  Move 1 from A to B
  Move 2 from A to C
  Move 1 from B to C
- To move N disks from A to C and we already know how to move N-1 disks from any one peg to another:
  1. Move the top N-1 disks by a series of legal moves from A to B using C. That leaves the largest disk (Disk N) in peg A.
  2. Move Disk N from A to C directly
  3. Move the N-1 disks on peg B by a series of legal moves from B to C using A

Algorithm

This is a recursive problem that can be solved by the following recursive algorithm:

```
Towers (N, Source , Target , Aux)
{
    if (N == 1) move disk 1 from Source to Target directly
    else
    {
        Call Towers to move N-1 disks from Source to Aux via Target
        Move disk N from Source to Target directly
        Call Towers to move N-1 disks from Aux to Target via Source
    }
}
```

An animation of the game is available at:

http://mathworld.wolfram.com/TowerofHanoi.html

Since any single disk move is always from or to the top of the peg, it is natural to represent the pegs with their disk contents as stacks of disks.

Required Implementations:

1. Implement an array-based stack template class ***Stackt.***
2. Develop a program using the algorithm given above to simulate the Towers of Hanoi game. Number the disks 1,2,3,…,N in ascending order of their size. Using the ***Stackt*** class, assign a ***stack*** to each peg to represent its disk content at any moment. Display the stacks to see each move until all disks have been moved from peg A to peg C. For a given N, display the number of moves needed.

Problem 2: A Simple Calculator Program (50 points)

Create a program that will read an infix expression from the user, convert the infix expression into an equivalent postfix expression, then evaluate the postfix expression. To make the problem simpler, use the following specifications:

- Operands are single digit integers only
- Infix expressions should support parentheses '(' and ')' and only the arithmetic binary operators of + , - , * , /
- Infix expressions should follow the usual C++ precedence rules
- Use a stack of characters for the conversion from the infix string to the postfix string, and a stack of *double* for the evaluation of the postfix expression

Your output should show:

1. The input infix string,
2. The converted postfix string,
3. The result of calculation.

For example:

Type your infix expression:
(5+3)(6-4)/((4-2)*(3+1))*
*Converted Postfix string is: 53+64-*42-31+*/*
Result is 2.0

Bonus (3 points) if your program can handle multiple digit integers, e.g., 23+30*4/15
