CMPT 225, Spring 2018, Assignment 3

Due date: February 20, 2018, 5:30 PM (before class)

1. Programming exercise. Implement a program that checks whether a given input string is a palindrome (i.e. is identical to itself if read from right to left), using both a stack and a queue. **(20 pts.)**

   Details and suggestions:

   You may use either an array-based or a linked list-based implementation of stacks and queues.

   Your algorithm may only use those operations that are specified in the queue and stack ADTs.

   Provide an implementation of your algorithm in a file called CheckPalindrome.cpp.

   The main function should accept an input string and print out **true** or **false**.

   Test cases:

   On input

   "racecar"

   your program should print out

   true.

   On input

   "gohangasalamiimalasagnahog"

   your program should print out

   true.

   On input

   "notapalindrome"

   your program should print out

   false.

2. Programming exercise. Implement a stack-based program for evaluating arithmetic expressions containing the standard arithmetic operators $+, -, *, /$, as well as $<=$ and $>=$. The precedence of the operators is the standard one: $*, /$ are at the same level of precedence, which is higher than the precedence for $+, -$ (same level), which itself is higher than the precedence for $<=, >=$ (also same level). **(30 pts.)**

   Details and suggestions:

   Use two stacks, one containing the operators and the other containing the values.

   You may use the **stack** class in the C++ Standard Library or design your own.

   The algorithm is described in detail in the lecture notes for Lecture 5.

   The input is a string of alternating integers and operators, separated by one space.

   The first and the last element of the input will be a number.

   You may assume that the expression contains at most 1 operator of type $<=$ or $>=$.

   If it does, it evaluates to either **true** or **false**; otherwise, it evaluates to a number.

   You may use the *strtok* command to get the operands contained in the input string.

   Provide an implementation of your algorithm in a file called EvaluateExpression.cpp.

   The main function should accept the string, parse it, and print out the value it has.

   Bonus: test the input string and throw the *BadExpression* exception if needed.

   Test cases (you must make sure your program works on these):

   On input

   *"1 + 2 - 3 * 4 / 5"*

   your program should print out

   0.6.

   On input

   *"12 - 3 * 2 + 7"*

   your program should print out

   13.

   On input

   *"14 <= 4 - 3 * 2 + 7"*

   your program should print out

   false.

   If you are attempting the bonus question, on input

   *"14 <= 4 - 3 * 2 + 7 <= 16"*

   your program should

   throw the *BadExpression* exception.

3. Suppose an initially empty stack S has performed a total of 25 *push* operations, 12 *top* operations, and 10 *pop* operations, 3 of which generated a *StackEmpty* exception without changing S. What is the current size of S? Explain your answer. **(5 pts.)**

4. Give a recursive function for removing all the elements in a stack. **(10 pts.)**

5. Describe, in pseudocode, how you can implement all the functions of the queue ADT using two stacks. Do not assume a specific implementation of the stacks. **(15 pts.)**

6. Describe, in pseudocode, how you can implement all the functions of the stack ADT using two queues. What is the running time of the *push* and *pop* functions in this case? **(20 pts.)**

7. **Bonus:** Prove the correctness of the algorithm seen in class for computing the spans of an array using a stack. You may want to use strong induction. **(5 pts.)**