Week 4 Exercises Submit

Sandra Batista

1.1–1.2

1. Queue Experimental Runtime

We have provided you with two implementations of integer Queue. You will study those implementations and determine which is more efficient and why.

This is problem 0 from HW2:
https://bytes.usc.edu/cs104/homework/hw2/

The code is also available here:
https://github.com/sandraleeusc/csci104_fall2020_lecture

2. isParenthesized() function

Write a function, isParenthesized, that given an expression string will return true if and only if the string is properly parenthesized and false otherwise. Recall from lecture and hw1 that an expression is properly parenthesized if a left parentheses such as '(' and '[' have a matching right parentheses. Symbols such as '+', '*', and numbers may be ignored. You may assume expression strings will only contain '(', ')", '[', ']', '+', '*', or numbers. Your implementation must be iterative. You may use STL container classes.

bool isParenthesized(string exp);
/** isParenthesized("(7+8)") and isParenthesized("[7*(6+8)]") should return true **/
/** isParenthesized("[7+8") and isParenthesized("[7+8)") should return false **/
/** isParenthesized("(7+[8)-5]") should return false **/
/* You may begin your implementation here*/

(Yes this is the same exercise as HW1)

3. Evaluate expression

Write a function, evaluate, that given a string containing a mathematical expression that is properly parenthesized and has whitespace between all tokens in the expression will evaluate the expression and return the value of the expression. string will evaluate the mathematical expression and return a double for the value. The strings will only contain “(“, “)”, “+”, “-”,”*”,”/”, and numbers represented as doubles. (For the numbers they may be given with or without a decimal point, e.g. 5 or 5.0)

double evaluate(string exp);
/** evaluate(“((5 * 4) * (2 + 3))”) should return the value 100.0 **/
/** evaluate(“((1 + (5 * (3 + 1))) / 4)”) should return the value 5.25 **/

// Hint 1: It may be preferable to use two stacks for this problem.
// One for the operand strings and another for the numbers.

// Hint 2: This may be a good time to practice using stringstreams.

4. Doubly Linked List

For the double linked list class, DoubleLL available here:
https://github.com/sandraleeusc/csci104_fall2020_lecture

Files: DoubleLL.h, DoubleLL.cpp and driver double_test.cpp

We have the push front we discussed as group, but the double_test.cpp has memory leaks because no destructor is implemented.

Write and submit

DoubleLL::~DoubleLL();

void DoubleLL::remove(std::shared_ptr<Item> p);