Homework 5

Due 20 February 2022, 11:59pm

**Name**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

**What to submit**:

* For text response:

Create a pdf file that shows all your answer for problems that requires text response. Upload this file to Gradescope.

* For programming:

For each programming question, run `make submission` to generate a zip file and upload it into Canvas.

**Questions**:

1. (programming) Use the provided starter code to convert an expression in postfix notation to prefix notation. You only have to add to the section commented "//TODO". Note that the expression should be purely symbolic with no numbers. (Ex: using 32+ is invalid). You are not required to submit any test cases; however, you should test your code by running the program postpre with different expressions from the terminal to see if the output is correct.

POSTFIX Grammar:

<postfix> = <operand> | <postfix> <postfix> <operator>

<operator> = ‘+’ | ‘-’ | ‘\*’ | ‘/’

<operand> = ‘a’|’b’|’c’| … |’z’|

Hint: The textbook has a pseudocode of the inverse of this function (converting prefix to postfix). Check this out as it will help you in understanding what should be done for our function here.

* 1. Use the provided starter code in homework/hw05/pospre.
  2. algebraic\_expressions.hpp and algebraic\_expressions.cpp have the functions headers and definitions needed. You only have to implement the section commented "//TODO".
  3. postpre.cpp is the main file that you should build. After you run cmake and make commands, use the following notation to run the program:   
       
     ≫ ./postpre <arg>  
       
     Where arg is the postfix expression you want to convert (e.g. ./postpre ab+)

1. (text response) Write an implementation of the ADT stack that uses a resizable array to represent the stack items. Anytime the stack becomes full, double the size of the array. Maintain the stack’s bottom entry at the beginning of the array.
   1. Add a feature that the array can also be decreased in size by half if the stack entries drop to less than half the size of the array. (Note: The array size should never drop below the original allocated size.)
   2. Add a flag *h*. If *h* is set, the array size should double if more than half of the entries are consumed. Otherwise, if it is not set, it should revert to the original method for growing.
   3. Add a second flag called *k*. If *k* is positive and if *h* is not set, each time the stack becomes full, increase the size of the array by the next value in the sequence 3*k,* 5*k*, 7*k*,….

Your answer.

Extra Pages

[Write “To Be Continued” in the above answer box if you use this extra space]

Extra Pages

[Write “To Be Continued” in the above answer box if you use this extra space]