\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**CSC130 Computing Fundamentals II**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

LAB 12B **Algorithm Design, Analysis and Time Complexity**

# Objectives

- Design and analyze algorithms

- Evaluate programs in terms of time complexity

# Instruction and Problems

## Part 1: Program Evaluation and Time Complexity

A palindrome is a sequence of characters that reads the same as the sequence in reverse. For example, ‘noon’ and ‘level’ are palindromes, while ‘moon’ and ‘lever’ are not. The function isPalindrome function determines whether a string is a palindrome or not.

def isPalindrome(string):

"""Returns True if string is a palindrome or False otherwise."""

stk = ArrayStack()

for ch in string:

stk.push(ch)

for ch in string:

if ch != stk.pop():

return False

return True

1. What is the time complexity of this program?

Answer:

O(n)

1. Justify your answer in part (a) by providing an analysis of the program.

Answer:

Each for loop will execute at a constant time of n. Even though there are two they are sequential, and we can ignore the statements and if statement since it’s a lower order, resulting in a time complexity of O(n).

Part 1 Rubric:

1. Time complexity [10 points]
2. Algorithm analysis [10 points]

## Part 2: Algorithm Design and Analysis

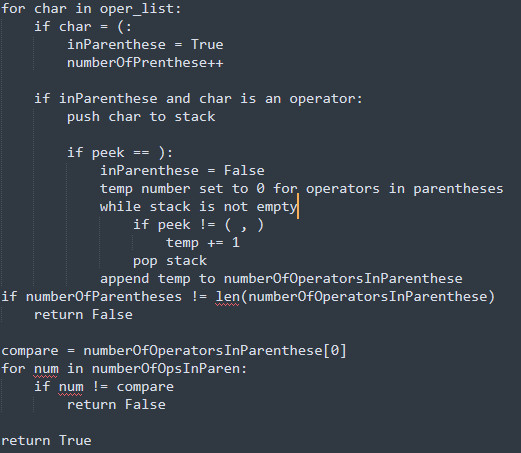
Parentheses are used in mathematical expressions to denote modifications to normal order of operations. They must be used in pairs and can be nested inside another pair. For example, parentheses in this expression ((5-2)\*(3+6))\*4 are matched properly, while parentheses in this expression ((2\*(3+6)+4)\*5 are not.

Develop an algorithm for a function parenthesesMatched, which tests whether parentheses in a mathematical expression are matched properly. This function has a parameter oper\_list, which is a list that holds the operands and operators of a mathematical expression. For example, the oper\_list for the expression ((5-2)\*(3+6))\*4 is ['(', '(', '5', '-', '2', ')', '\*', '(', '3', '+', '6', ')', ')', '\*', '4']. This function returns True if the parentheses are matched properly, and False otherwise. [Hint: Consider using a stack]

1. Design the algorithm and express it in pseudocode.

Answer:

In the pseudocode provided below, I begin with a for loop of the passed in oper\_list and check for the first opening parenthesis. Once found, set a Boolean value to true to indicate we are in a parenthesis and to push the incoming operators to the stack. Once a closing parenthesis is reached enter a while loop to extract the number of operators in the parentheses and append it to a list. At the end of the oper\_list we check if the length of the operators list is equal to the number of parentheses found in the equation, and that the number of operators is the same within each parenthesis. Returning true at the end if all conditions are met.



1. What is the time complexity of your algorithm? Justify your answer by providing an analysis of your algorithm.

Answer:

I am stuck between O(n) and O(n^2) for this algorithm. The while loop nested inside the for loop is making me question whether which one is correct. This is because the while loop is not checking each element of the char array which is considered n in this case. If it was, we would multiple n\*n, but this isn’t the exact case. If I had to guess it would be O(n) since big O notation denotes the upper mathematical bounds and the while loop does not equate to a true O(n) in this case.

Part 2 Rubric:

1. Algorithm design [20 points]
2. Algorithm analysis [10 points]