# Computational complexity of question 4

Lets look at the function doBracketsMatch line by line

Line 1: let stack = new BracketStack()

If we take a look at the BracketStack object, it does not contain an init method nor a loop that is run when creating the object. It only contains a line that initializes the variable openBracketsCount Therefore, this line will be constant time. We can denote this as O(1)

Line 2: let isOpeningSymbol = isSymbol(openingSymbol)

The isSymbol object only executes one line when it is created. Therefore has a constant time. Thus, this line is of constant time O(1)

Line 3: let isClosingSymbol = isSymbol(closingSymbol)

The same reasoning in line 2 applies here. So we can denote this also as O(1)

Line 4: for (let I = 0; I < inputString.length; i++)

This for loop has a runtime of n, where n is the length of inputString. This line has a time O(n)

Line 5: let value = inputString[i]

This line has a constant runtime. Denote it as O(1)

Line 6: if (isOpeningSymbol(value)) stack.push()

This is a simple if statement, therefore requires a constant time. We can denote it as O(1)

Line 7-9:

If (isClosingSymbol(value))

If (stack.isEmpty()) return false

Else stack.pop()

This is also a simple nested if statement. At worse case, the time is O(2)

Line 11: return stack.isEmpty()

The function isEmpty is a simple if statement. Therefore has O(1). The incrementing of the variable also is O(1). Therefore this line is O(2).

The total time Tn = 3\*O(1) + O(n)\*(2\*O(1) + O(2)) + O(2)

In computational complexity we can drop all the constant terms. Thus the time/computational complexity of the function doBracketsMatch is O(n).