Question statement (2):

Write a function that takes as input a string and returns whether the parenthesis are "balanced".

If the candidate solves this question quickly, add the following difficulty: the string may also contain "[]" or "{}". Return whether all three types of brackets are balanced.

So 423

Question Statement (2)

I - dentify the problem

figure out if there are extra parentheses in a string

D - efine the goal

-) essentially count the parentheses
  -) make like account for them

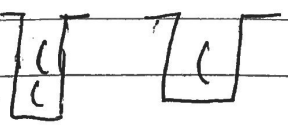
-) the output is a boolean bc:
  we want to know whether or not the parens are balanced.

E - xplore
A - nticipate
Done using Duke's Approach

I) Examples:

-) ex: "((a+b)-c)" In normal situations how will things operate?

// it's a string so I have to keep track of indices



-) push when we see an opening
-) pop when we see a closing

what happens if string starts w/ a closing

·) ex: ")(("

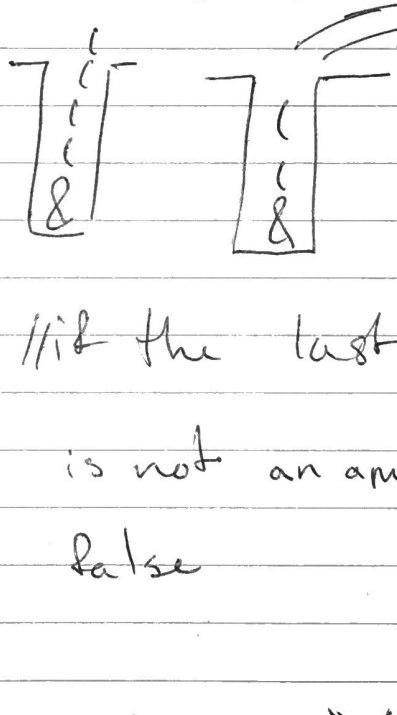
// can't pop when we have an

✓ // empty ~~string~~ stack => we ALWAYS

// push an indicator char before

// reaching an empty stack

·) ex: "((((()" what happens w/ more opening than closing

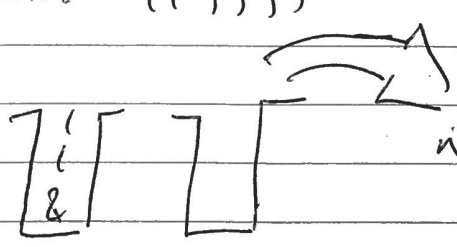


//if the last thing popped

is not an ampersand return

false

·) ex: "(()))" what happens w/ more closing than opening



// if the stack is not empty

↑

while

pop the chars. If it's empty before the ampersand return false

what happens w/ other chars?

- ex: "(xyzABC)&"

// one extra pop to see at the end of string iter

// what's remaining

// ignore all other chars

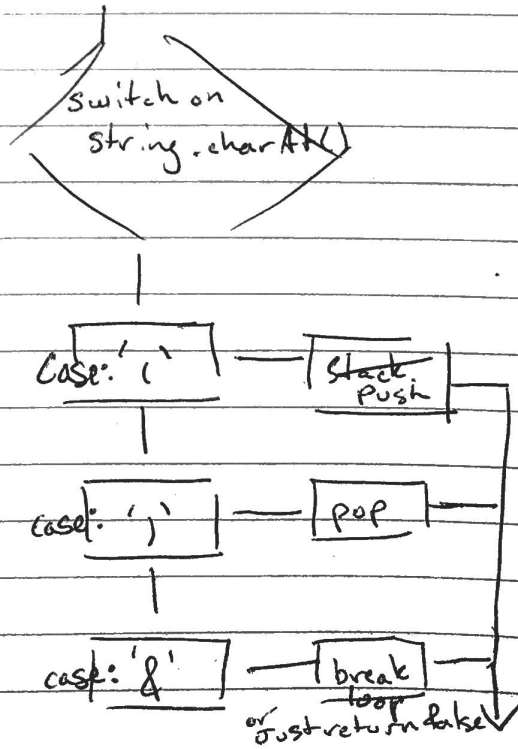
- ex: "(123(a)"

5) translate to code

1st) check for empty string

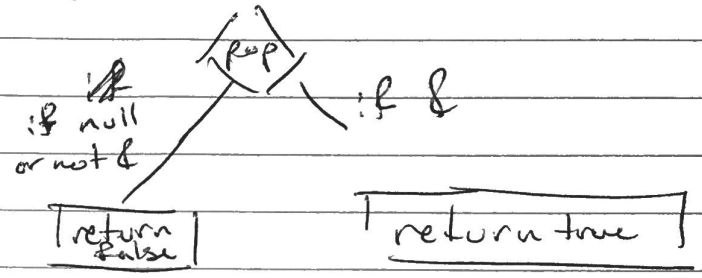
2nd) push '&' to stack

3rd) check while not at end of string



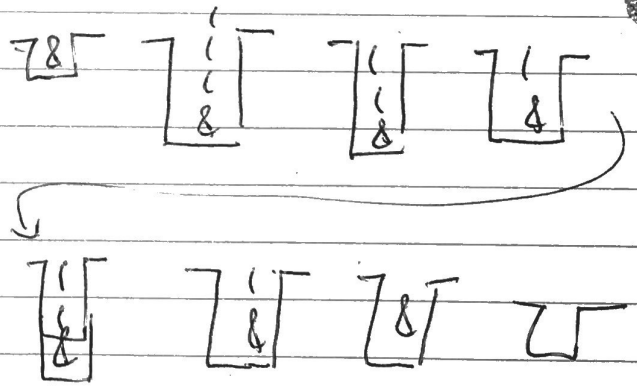
4th) when end of string

~~pop to check stack~~



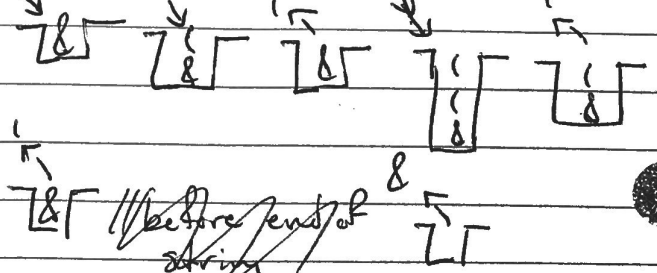
4) check by hand

·) ex: "(((a+b)/c)+(d/e))+&"



return true

·) ex: "()+((a+b)/c))"



one more pop for final check returns null