A string S consisting of N characters is called *properly nested* if:

* S is empty;
* S has the form "(U)" where U is a properly nested string;
* S has the form "VW" where V and W are properly nested strings.

For example, string "(()(())())" is properly nested but string "())" isn't.

Write a function:

def solution(S)

that, given a string S consisting of N characters, returns 1 if string S is properly nested and 0 otherwise.

For example, given S = "(()(())())", the function should return 1 and given S = "())", the function should return 0, as explained above.

Write an **efficient** algorithm for the following assumptions:

* N is an integer within the range [0..1,000,000];
* string S consists only of the characters "(" and/or ")".

def solution(S):

if len (S) == 0:

return 1

if len (S) % 2 != 0:

return 0

arr1 = []

for i in S:

if len(arr1) == 0:

arr1.append(i)

elif i ==")" and arr1[len(arr1)-1] =="(":

arr1.pop( len(arr1)-1 )

else:

arr1.append(i)

if len(arr1) == 0:

return 1

else:

return 0

A = '())(()'

car1 = solution(A)

print(car1)