**DAY-4**

'''

An valid bracket arrangement is either "", "[" + A + "]",

or A + B, where A and B are valid bracket arrangements

and + represents string concatenation.

For example, "", "[]", "[[]]", and "[[][[]]]" are all valid bracket combinations.

A non-empty valid bracket arrangement is called authentic bracket string.

If an authentic bracket string s can be divided into s = A + B, where A and B

are valid bracket strings and non-empty.

Consider the following basic partitioning of an authentic bracket string S,

S = P1 + P2 +... + Pk, where Pi are basic authentic bracket string.

Return S after deleting the outermost brackets of each authentic string in

S's partitioning.

INPUT FORMAT:

-------------

A String S, consists of '[' and ']' brackets only.

OUTPUT FORMAT:

--------------

Print S after deleting the outermost brackets.

SAMPLE INPUT-1:

---------------

[[]][[][]][]

SAMPLE OUTPUT-1:

----------------

[][][]

SAMPLE INPUT-2:

---------------

[[][]][[]][[][[]]]

SAMPLE OUTPUT-2:

----------------

[][][][][[]]

Write your Python code below:

'''

Soln

n=list(input())

count=0

start=0

for i in range(0,len(n)):

if(n[i]=='['):

if(count==0):

start=i

count+=1

else:

count-=1

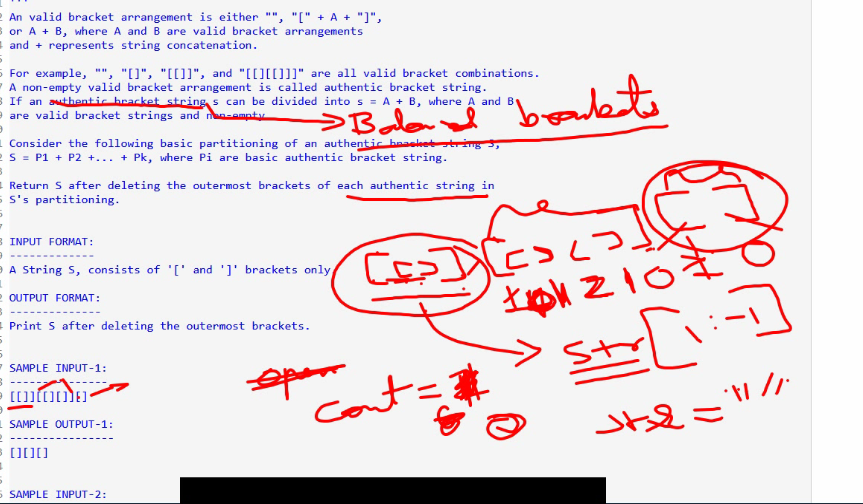
if count==0:

n[i]=""

n[start]=""

output=''.join(n)

print(output,end="")



You have been tasked with managing scores in a unique game.

The game begins with an empty scorecard. You are provided with a list

of operations represented as a string of characters, denoted as operations[i],

each corresponding to a specific action. These actions include:

A digit S: This means you should add a new score of S to the scorecard.

'A': This means you should add a new score to the scorecard,

which is the addition of the last two scores.

'D': This means you should add a new score to the scorecard,

which is double the value of the previous score.

'R': This means you should invalidate the previous score,

removing it from the scorecard.

Your objective is to calculate the sum of all the scores that remain on

the scorecard after performing all the specified operations.

Input Format:

-------------

A string consists of characters, represents series of operations.

Output Format:

--------------

An integer result.

Sample Input-1:

---------------

526RDAA

Sample Output-1:

----------------

27

Explanation:

-----------

'5' - Add 5 to the scorecard, scorecard is now [5].

'2' - Add 2 to the scorecard, scorecard is now [5, 2].

'6' - Add 6 to the scorecard, scorecard is now [5, 2, 6].

'R' - Invalidate and remove the previous score, scorecard is now [5].

'D' - Add 2 \* 2 = 4 to the scorecard, scorecard is now [5, 2, 4].

'A' - Add 2 + 4 = 6 to the scorecard, scorecard is now [5, 2, 4, 6].

'A' - Add 4 + 6 = 10 to the scorecard, scorecard is now [5, 2, 4, 6, 10].

The total sum is 5 + 2+ 4 + 6 + 10 = 27.

Sample Input-2:

---------------

1R

Sample Output-2:

----------------

0

Soln:

import java.util.\*;

public class Main{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

ArrayList<Integer> k=new ArrayList<>();

String s=sc.next();

for(int i=0;i<s.length();i++){

if(s.charAt(i)!='A'&& s.charAt(i)!='D' && s.charAt(i)!='R'&& s.charAt(i)!='S'){

k.add(Integer.parseInt(String.valueOf(s.charAt(i))));

}

if(s.charAt(i)=='A'){

// if(i-1>0 && i-2>=0 ){

int sum=k.get(k.size()-1)+k.get(k.size()-2);

k.add(sum);

// }

}

if(s.charAt(i)=='D'){

k.add(2\*k.get(k.size()-1));

}

if(s.charAt(i)=='R'){

k.remove(k.size()-1);

}

}

int sum=0;

for(int i=0;i<k.size();i++){

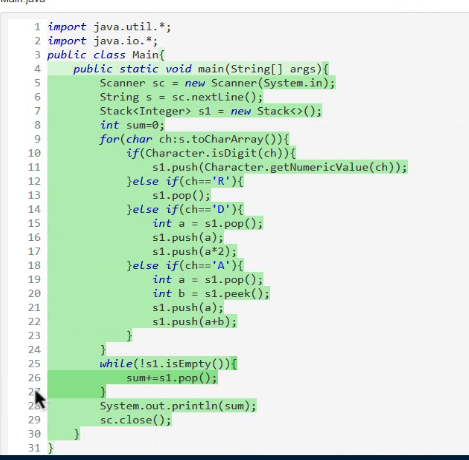
sum+=k.get(i);

}

System.out.println(sum);

}

}



Ram and Bheem are using a Desktop Computer.One day they found that keyboard

is defective in which if you type backspace button,it will print '$',

instead of removing one previous character.

Bheem and Ram have tried to type one word each on the same keyboard.

Return true, if both tried to type the same word. Otherwise return false.

Note:backspace for an empty text will continue empty.

Input Format:

-------------

Line-1:Two space seperated strings represents words w1,w2.

Output Format:

--------------

Print a boolean result.

Constraints:

1 <= w1.length, w2.length <= 200

w1 and w2 only contain lowercase letters and '$' characters.

Sample Input-1:

---------------

pq$r pt$r

Sample Output-1:

----------------

true

Explanation:

------------

Both wants to type 'pr'

Sample Input-2:

---------------

se$$at cea$$t

Sample Output-2:

----------------

false

Sample Input-3:

---------------

s$$at ce$$at

Sample Output-2:

----------------

true

Explanation:

------------

Both wants to type 'at'.

Soln:-**this is my soln and is optimized**

import java.util.\*;

public class Main{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

String s1=sc.next();

String s2=sc.next();

Stack<Character> s11=new Stack<>();

Stack<Character> s22=new Stack<>();

for(char i:s1.toCharArray()){

if(i=='$'){

if(s11.isEmpty()){

continue;

}

else{

s11.pop();

}

}

else{

s11.push(i);

}

}

for(char i:s2.toCharArray()){

if(i=='$'){

if(s22.isEmpty()){

continue;

}

else{

s22.pop();

}

}

else{

s22.push(i);

}

}

s1="";s2="";

for(Character i: s11){

s1+=i;

}

for(Character i: s22){

s2+=i;

}

if(s1.equals(s2)){

System.out.println(true);

}

else{

System.out.println(false);

}

}

}