Codility_

CodeCheck Report: training84ZGZR-DJV

Test Name:

Check out Codility training tasks

Summary Timeline

Tasks summary

Task	Time spent	Score
Nesting Java 8	2 min	100%

Total score

Tasks Details

1. Nesting

Easy

Determine whether a given string of parentheses (single type) is properly nested.

Task Score	Correctness	Performance
100%	100%	100%

Task description

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:

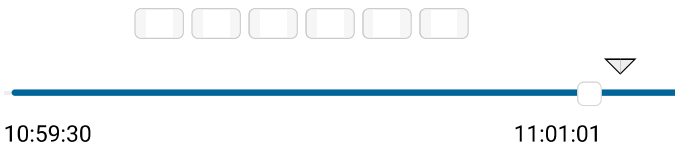
```
class Solution { public int solution(String S); }
```

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

Solution

Programming language used:	Java 8
Total time used:	2 minutes
Effective time used:	2 minutes
Notes:	*not defined yet*

Task timeline



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 ")".

Code: 11:01:00 UTC, java, final, score: **100**

show code in pop-up

```
// you can also use imports, for example:
// import java.util.*;
import java.util.ArrayList;
// you can write to stdout for debugging purpo
// System.out.println("this is a debug message

class Solution {
    static ArrayList<String> bracketStack = ne

        public static void push(ArrayList<Stri
                stringList.add(s);
        }

        public static String pop(ArrayList<Str
                return (stringList.isEmpty() ?
        }
    public int solution(String S) {
        // write your code in Java SE 8

        int output = -1;
                boolean isNestedString = true;
                if(!S.isEmpty()) {
                        int index = 0;
                        int size = S.length();
                        boolean continueWhile
                        do {
                                //char current
                                String current

                                switch (curren
                                case ")":
                                        if (!p


                                        }
                                        break;
                                case "}":
                                        if (!p


                                        }
                                        break;
                                case "]":
                                        if (!p


                                        }

                                        break;

                                default:
                                        push(b
                                        break;
                                }

                                index++;
                        } while (index<size&&c

                }

                if(isNestedString&&bracketStac
                        output = 1;
                }else {
                        output = 0;
                }

                return output;
    }
}
```

Analysis summary

The solution obtained perfect score.

Analysis

Detected time complexity: **C**

expand all	Example tests
▶ example1 example test	
▶ example2 example test2	
expand all	**Correctness tests**
▶ negative_match invalid structure, but the number of parentheses matches	
▶ empty empty string	
▶ simple_grouped simple grouped positive and negative test, length=22	
▶ small_random	
expand all	**Performance tests**
▶ large1 simple large positive and negative test, 10K or 10K+1 ('s followed by 10K)'s	
▶ large_full_ternary_tree tree of the form T=(TTT) and depth 11, length=177K+	
▶ multiple_full_binary_trees sequence of full trees of the form T=(TT), depths [1..10..1], with/without unmatched ')' at the end, length=49K+	
▶ broad_tree_with_deep_paths string of the form (TTT...T) of 300 T's, each T being '(((...)))' nested 200-fold, length=1 million	