Codility_

CodeCheck Report: training4JU5ZC-X5B

Test Name:

Check out Codility training tasks

Summary | Timeline

Mail Status: Not Applicable

- Finished: 2021-10-22 00:29 UTC
- Started: 2021-10-22 00:20 UTC
- Invitation Created: 2021-10-22 00:20 UTC

Tasks Details

Easy

1. **Nesting**
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:

```
int solution(string &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 ")".

Solution

Programming language used:	C++
Total time used:	9 minutes
Effective time used:	9 minutes
Notes:	*not defined yet*

Task timeline

00:20:55 00:29:02

Code: 00:29:01 UTC, cpp, final, score: **100** show code in pop-up

```
1  // you can use includes, for example:
2  // #include <algorithm>
3
4  // you can write to stdout for debugging purpo
5  // cout << "this is a debug message" << endl;
6
7  int solution(string &S) {
8     int cont=0,n=S.size();
9     for(int i=0;i<n;i++){
10        if(S[i]=='(' || S[i]=='{' || S[i]=='['
11            cont++;
12        }
13        else if(S[i]==')' || S[i]=='}' || S[i]=
```

```
14              cont --;
15              if(cont<0){
16                  return 0;
17              }
18          }
19      }
20      if(cont!=0){return 0;}
21      else{return 1;}
22  }
```

Analysis summary

The solution obtained perfect score.

Analysis

Detected time complexity: **O(N)**

expand all	Example tests	
▶	example1 example test	✔ OK
▶	example2 example test2	✔ OK
expand all	**Correctness tests**	
▶	negative_match invalid structure, but the number of parentheses matches	✔ OK
▶	empty empty string	✔ OK
▶	simple_grouped simple grouped positive and negative test, length=22	✔ OK
▶	small_random	✔ OK
expand all	**Performance tests**	
▶	large1 simple large positive and negative test, 10K or 10K+1 ('s followed by 10K)'s	✔ OK
▶	large_full_ternary_tree tree of the form T=(TTT) and depth 11, length=177K+	✔ OK
▶	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+	✔ OK
▶	broad_tree_with_deep_paths string of the form (TTT...T) of 300 T's, each T being '(((...)))' nested 200-fold, length=1 million	✔ OK