Codility_

CodeCheck Report: training4JU5ZC-X5B

Test Name:

Check out Codility training tasks

Summary Timeline

Tasks summary

Task	Time spent	Score
Nesting C++ ⚠	9 min	100%

Total score

100%

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

```
// you can use includes, for example:
// #include <algorithm>

// you can write to stdout for debugging purp
// cout << "this is a debug message" << endl;

int solution(string &S) {
   int cont=0,n=S.size();
   for(int i=0;i<n;i++){
       if(S[i]=='(' || S[i]=='{' || S[i]=='['
           cont++;
       }
       else if(S[i]==')' || S[i]=='}' || S[i]=
           cont --;
           if(cont<0){
               return 0;
           }
       }
   }
   if(cont!=0){return 0;}
   else{return 1;}
}
```

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