codility

Check out Codility training tasks

Candidate Report: Anonymous

Test Name:

Summary Timeline

Test Score

100 out of 100 points

100%

Tasks in Test

	Time Spent	Task Score
LongestPassword Submitted in: C++	1 min	100%

TASKS DETAILS

EASY

1. **LongestPassword**
Given a string containing words, find the longest word that satisfies specific conditions.

Task Score	Correctness	Performance
100%	100%	Not assessed

Task description

You would like to set a password for a bank account. However, there are three restrictions on the format of the password:

- it has to contain only alphanumerical characters (`a−z`, `A−Z`, `0−9`);
- there should be an even number of letters;
- there should be an odd number of digits.

You are given a string S consisting of N characters. String S can be divided into *words* by splitting it at, and removing, the spaces. The goal is to choose the longest word that is a valid password. You can assume that if there are K spaces in string S then there are exactly K + 1 words.

Solution

Programming language used:	C++
Total time used:	1 minutes
Effective time used:	1 minutes
Notes:	*not defined yet*

Task timeline

For example, given "`test 5 a0A pass007 ?xy1`", there are five words and three of them are valid passwords: "`5`", "`a0A`" and "`pass007`". Thus the longest password is "`pass007`" and its length is 7. Note that neither "`test`" nor "`?xy1`" is a valid password, because "`?`" is not an alphanumerical character and "`test`" contains an even number of digits (zero).

Write a function:

```
int solution(string &S);
```

that, given a non-empty string S consisting of N characters, returns the length of the longest word from the string that is a valid password. If there is no such word, your function should return −1.

For example, given S = "`test 5 a0A pass007 ?xy1`", your function should return 7, as explained above.

Assume that:

- N is an integer within the range [1..200];
- string S consists only of printable ASCII characters and spaces.

In your solution, focus on **correctness**. The performance of your solution will not be the focus of the assessment.

18:39:48 18:40:22

Code: 18:40:22 UTC, cpp, final, score: **100**

show code in pop-up

```
#include<string>
#include <algorithm>

// Should be alphanumeric
// Should contain even number of letters
// Should contain odd number of digits

bool IsAlphaNumeric(const std::string& str) {
   bool result = std::all_of(str.begin(), str.end(
   return result;
}

bool HasEvenNumberOfLetters(const std::string& str
   int count = std::count_if(str.begin(), str.end(
   return (count % 2 == 0);
}

bool HasOddNumberOfDigits(const std::string& str)
   int count = std::count_if(str.begin(), str.end(
   return (count % 2 != 0);
}

constexpr static char SPACE = ' ';
std::string GetNextPassword(std::string& str, size
   size_t nextspace = str.find(SPACE, curpos);
   std::string result;

   if (nextspace != std::string::npos) {
      result = str.substr(curpos, nextspace - curp
   }
   else if (curpos < str.length()) {
      result = str.substr(curpos, str.length() - c
   }
   return result;
}

bool IsValidPassword(const std::string& str) {
   if (IsAlphaNumeric(str) && HasEvenNumberOfLette
      return true;
   }
   return false;
}

int solution(std::string& S) {
   std::string password;
   int maxlen = 0;
   size_t offset = 0;
   bool validpasswordpresent = false;
   do {
      password = GetNextPassword(S, offset);
      size_t len = password.length();

      if (IsValidPassword(password)) {
         validpasswordpresent = true;
```

```
            if (len > maxlen) {
                maxlen = len;
            }
        }
        offset += (len + 1);
    } while (!password.empty());
    return validpasswordpresent ? maxlen: -1;
}
```

Analysis summary

The solution obtained perfect score.

Analysis

expand all	Example tests
▶ example example test	✓ OK
expand all	**Correctness tests**
▶ simple short and simple tests	✓ OK
▶ one_character one character words	✓ OK
▶ one_word tests that contains one word only	✓ OK
▶ even_letters all words have even number of letters	✓ OK
▶ odd_digits all words have odd number of digits	✓ OK
▶ odd_length it's sufficient to test validity of characters and if length of word is odd	✓ OK
▶ all_alphanumerical all words contain only alphanumerical characters	✓ OK
▶ extra_characters valid passwords joined with some invalid characters	✓ OK
▶ large_random random tests	✓ OK
▶ maximum biggest possible tests with mixed types of words	✓ OK