GTU Department of Computer Engineering
CSE 222/505 - Spring 2023
Homework 4
Due date: April 16, 2023 – 23:59

SCENARIO

Assume that you are designing a security system for the museum in Topkapı Palace. The museum security system must provide an encrypted password for each museum officer. When the officers need to enter the warehouse where the most valuable items of the museum are stored, they need a personal password pair to type to the security system which is in front of the entrance of the warehouse. The personal password pair is specified and assigned to the officer. **When an officer types their username and personal password pair, your system is responsible for decoding the passwords to check if they are valid or not.** If the passwords are valid, the system opens the entrance door to let the officer get in. Accept that the passwords have already been given to the officers by the manager of the museum. Your security system does not need to generate the passwords of the officers.

OBJECTIVE

When an officer types their username and personal password pair, your system is responsible for decoding the passwords to check if they are valid or not.

INPUT

The officer types three following inputs to the security system. A username and a pair of passwords for each officer. The inputs can be defined as static variables in the main class.

- **username (Data type: String):** contains only letters. The minimum length of the string is 1 (and can be more than 1).
- **password1 (Data type: String):** contains only letter and brackets (‘{‘, ‘}’, ‘[‘, ‘]’, ‘(‘, ‘)’). Each string password must contain at least 2 brackets. The minimum length of the string must be 8, including the brackets (and can be more than 8).
- **password2 (Data type: Int):** The number must be between 10 and 10000.

ENCRYPTION

Your system has the following functions which decode one of the inputs to check if it is valid or not.

1. **[A Recursive Function]** boolean checkIfValidUsername(String username): a function which checks if it contains only letters, and the minimum length is 1.
2. **[A Stack Function]** boolean containsUserNameSpirit(String username, String password1): a function which checks if the string password contains at least one letter of the username. For example:
    a. Username= gizem, password1= "{[(abacaba)]}" (output: False)
    b. Username= gokhan, password1= "{[(abacaba)]}" (output: True)
3. **[A Stack Function]** boolean isBalancedPassword(String password1): In the given string sequence, the function considers two brackets to be matching if the first bracket is an open bracket, (ex: (, {, or [), and the next bracket is a closed bracket of the same type. String cannot start with a closed bracket. There can be letters between any two brackets. For example:
    a. password1 = "{[(abacaba)]}" (output: True)
    b. password1 = "{ab[bac]caba}" (output: True)
    c. password1 = ")abc(cba" (output: False)
    d. password1 = "a]bcd(cb)a" (output: False)
    e. password1 = "[a]bcd(cb)a" (output: True)

    Hint: You can remove the letters from the string password before computing the function. Ex: "{[(abacaba)]}" -> "{[()]}"

4. **[A Recursive Function]** boolean isPalindromePossible(String password1): In the given string sequence, the function considers if it is possible to obtain a palindrome by rearranging the letters in the string. The function ignores the brackets in the string while computing the function. While converting the string to palindrome; you cannot add/remove letters but you can rearrange them in the string. For example:
    a. password1 = "{[(ecarcar)]}" (output: True) ("racecar")
    b. password1 = "{ab[bac]aaba}" (output: False) ("Not possible")
    c. password1 = "{(abba)cac}" (output: True) ("abcacba")

    Hint: You can remove the brackets from the string password before computing the function. Ex: "{(abba)cac}" -> "abbacac"

5. **[A Recursive Function]** boolean isExactDivision(int password2, int [] denominations): Considering the given list of the denominations, the function determines if it is possible to obtain the password by the summation of denominations along with arbitrary coefficients, which are non-negative integers. For example:
    a. Password2 = 75, denominations = [4, 17, 29] (output: True)
    Explanation: 17+29+29=75. The coefficients are respectively 0, 1, and 2.
    b. Password2 = 35, denominations = [4, 17, 29] (output: False)
    c. Password2 = 54, denominations = [4, 17, 29] (output: True)
    Explanation: 4+4+17+29=54. The coefficients are respectively 2, 1, and 1.

    Note: The default list of denominations is [4, 17, 29] but the user may give a different list.

FINAL OUTPUT

If the given username and passwords get True from all five encryption functions, the final output of the system will be:

"The username and passwords are valid. The door is opening, please wait..."

Otherwise,

"The username is invalid due to (give the reason). Try again..."

"The string password is invalid due to (give the reason). Try again..."

"The integer password is invalid due to (give the reason). Try again..."

REPORT

- Give the time complexity analysis of each function in detail.
- Give the output results of your code with respect to the different inputs and explain your design.

GRADING

- No OOP design: -100
- No error handling: -50
- No javadoc documentation: -50
- No report: -30
- Disobey restrictions: -100
- Cheating: -200
- Your solution is evaluated over 100 as your performance.

IMPORTANT:

You are not allowed to use any library other than java.util.Stack.

CONTACT

Res. Asst. Sibel Gülmez: sgulmez2018@gtu.edu.tr