Strong Password Checker

The program takes a string *s* as input from the user and checks whether it is a strong password or not. It is considered to be a strong password if all three conditions are met:

- It has at least 6 characters and at most 20 characters.
- It must contain at least one lowercase letter, at least one uppercase letter, and at least one digit.
- It must NOT contain three repeating characters in a row ("...aaa..." is weak, but "...aa...a..." is strong, assuming other conditions are met).

With my approach in resolving this problem, the program offers the user the possibility to check multiple passwords in one session, until 0 is pressed to exit the program.

In the function *password_checker()*, first, it initializes the variables:

-*int n* (the length of the string),

-*bool lower*, *upper*, *digit* (all set to false, because we assume that the password does not contain least one lowercase letter, at least one uppercase letter, and at least one digit),

-*int repeat* (store the number of repeating consecutive characters),

-*int changes* (store the number of changes that we need to make to have a strong password, initialized with 0 because in the beginning there are 0 changes);

Then, it iterates over each character of the string using a for loop and storing it in a char variable *c*, and sets the Boolean variables *lower*, *upper*, *digit* to true if the character *c* matched the corresponding criteria.

The program checks for repeating characters by calling the function *count_characters(),* which returns the count of consecutive equal characters starting from the given index in the string. If the *count* is greater than or equal to 3, it means that it contains 3 repeating characters in a row which is a weak password, it adds the *count* divided by 3 to the *changes* variable(because if we

have 3 repeating characters we have to make only one change to make it a strong password, it is the same case when we have 4 or 5 repeating characters; but if we increase the repeating characters we will see that at each set of 3 we need exactly one change, therefore for 6,7,8 we need 2 changes, for 9,10,11 we need 3 changes and so on.

Afterwards, it calculates the minimum number of changes needed to make the password strong by adding up the changes needed to meet each criteria (i.e., adding 1 to changes for each criteria that is not met- digit, lower letter, upper letter), as well as any additional changes needed to meet the length requirement(at least 6 and at most 20).

Finally, it returns the variable *changes* that stores the number of minimum changes to make the given password a strong one, or if it is equal to 0 it means that it is already strong.

Examples:

```
Welcome to the Strong Password Checker!
Please input the password to be checked.
Press 0 if you want to exit.
>> 12strOng
The password is already strong!
Please input the password to be checked.
Press 0 if you want to exit.
>> pass1
MINIMUM change required to make s a strong password:2
Please input the password to be checked.
Press 0 if you want to exit.
>> passwordpasswordpassswprd123
MINIMUM change required to make s a strong password:10
Please input the password to be checked.
Press 0 if you want to exit.
>> paRolaaaaaaa
MINIMUM change required to make s a strong password:3
Please input the password to be checked.
Press 0 if you want to exit.
>> Paaaaasswoooord12
MINIMUM change required to make s a strong password:2
Please input the password to be checked.
Press 0 if you want to exit.
>> 0
Exiting the Strong Password Checker...
Process finished with exit code 0
```

12strOng → the password is strong because it has at least 6 characters, at most 20, it contains a lower case letter(ex: s), an upper case letter(ex: O), a digit(ex: 2), and does not have three repeating characters in a row.

pass1 → Minimum change is 2 because it has 5 characters(<6) so the user must add one more character(1 change) and it does not contain an upper case letter(1 change), the other requirements are fulfilled.

passwordpasswordpassswprd123 → Minimum change is 10 because it has 28 characters(>20) so the user must delete 8(8 changes), it does not contain an upper case letter(1 change) and it does have three repeating characters in a row(1 change), the other requirements are fulfilled.

paRolaaaaaaa → Minimum change is 3 because it has two sets of three repeating characters in a row(2 changes) and it does not contain at least one digit(1 change), the other requirements are fulfilled.

Paaaaasswoooord12 → Minimum change is 2 because it has 5 repeating characters “a” (1 change) and 4 repeating characters “o”(1 change), the other requirements are fulfilled.