**Problem4 – UMTSoftware**

A password is considered strong if below conditions are all met:

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

Write a function strongPasswordChecker(s), that takes a string s as input, and return the MINIMUM change required to make sa strong password. If s is already strong, return 0.

Insertion, deletion or replacement of any one character are all considered as one change.

Explanation of the problem:

The length problem. When the length is less than 6, I need to insert characters to supplement the length. When the length exceeds 20, I need to delete characters.

Missing characters or numbers. When I lack uppercase, lowercase and numbers, I can complete them by inserting characters or replacing characters.

I have calculated how many characters need to be added to the current password string to get to 6. The method of supplementing characters can only be operated by inserting characters, so when the number of missing types is less than or equal to diff, I don't need to add more operations. When diff cannot completely cover the number of missing types, I should also add the difference.

For the case where the number of repeated characters k is greater than or equal to 3, we do not directly delete it to 2, but first delete it to the nearest (3m +2), then if k is exactly divisible by 3, then we directly become k-1, and if k is divided by 3 with a remainder of 1, then it becomes k-2. The advantage of this is that 3m+2 repeated characters can most efficiently replace m characters to remove repetitions.

First, I calculated the number over of more than 20. I first added over to the result res, because the over deletion operation must be done anyway. If it does not exceed, over is 0, and the variable left represents the minimum number of replacements that need to be replaced to solve repeated characters, and is initialized to 0. Then I traversed the array that counts the number of characters before. If the number of a certain character is greater than or equal to 3, and at this time over is greater than 0, then we will reduce the number to the nearest 3m+2, and the over will also decrease accordingly. Note that once over is less than or equal to 0, do not delete anymore. If the number of all repetitions is reduced to 3m+2, but over is still greater than 0, then I need to further delete operations, this time directly delete 3m each time, until over is less than or equal to 0. If there are characters whose number of repetitions is greater than 3, I could calculate the number of replaced characters and add them directly to left. Finally, we compare left and missing, and take the larger value and add it to the result res.