UMT Internship – Tocaciu Valeria

In order to solve the problem, I have a dictionary where I keep all the changes that need to be made (the key is the name of the change, and the values is how many times that change needs to happen).

First I check the first condition, length of the password to see if I need to add or remove characters.

```
changes = {'addChr': 0, 'removeChr': 0, 'types': 0, 'repeatingChr': 0}
if len(password) < 6:
    changes['addChr'] = 6 - len(password)
elif len(password) > 20:
    changes['removeChr'] = len(password) - 20
```

Then I check the second one to see if the password contains lowercase and uppercase characters and digits.

```
lowercase = False
uppercase = False
digit = False
for letter in password:
    if letter.isupper():
        uppercase = True
    if letter.islower():
        lowercase = True
    if letter.isdigit():
        digit = True
if not lowercase:
    changes['types'] = changes['types'] + 1
if not uppercase:
    changes['types'] = changes['types'] + 1
if not digit:
    changes['types'] = changes['types'] + 1
```

In order to approach the third condition I noticed that there are 3 cases:

1. The length of the password is less than 6
2. The length of the password is between 6 and 20
3. The length of the password is more than 20

```
if changes['addChr'] != 0:
    return firstCase(password, changes)
if changes['addChr'] == 0 and changes['removeChr'] == 0:
    return secondCase(password, changes)
if changes['removeChr'] != 0:
    return thirdCase(password, changes)
```

For the first case if the characters if the number of characters that need to be added is greater than the “types” of characters missing then the second condition doesn’t matter. I check the third condition and for each occurrence of repeating characters, the characters that need to be added, and types are decreased, and the number of repeating characters increases (that means that in order for the password to be valid, a character needs to be added in that position). The result is the sum of the remaining values in the dictionary.

```python
def firstCase(password, changes):
    if changes['addChr'] >= changes['types']:
        changes['types'] = 0
    i = 0
    while i < len(password) - 2:
        if password[i] == password[i + 1] == password[i + 2]:
            if changes['addChr'] != 0:
                changes['addChr'] = changes['addChr'] - 1
            if changes['types'] != 0:
                changes['types'] = changes['types'] - 1
            changes['repeatingChr'] = changes['repeatingChr'] + 1
            i = i + 2
        i = i + 1
    return sum(list(changes.values()))
```

For the second case, I only check the types of the characters and the repeating sequences.

```python
def secondCase(password, changes):
    i = 0
    while i < len(password) - 2:
        if password[i] == password[i + 1] == password[i + 2]:
            if changes['types'] != 0:
                changes['types'] = changes['types'] - 1
            changes['repeatingChr'] = changes['repeatingChr'] + 1
            i = i + 2
        i = i + 1
    return sum(list(changes.values()))
```

And for the third case, the approach is similar to the first one, the only difference being that the “types” value gets decreased only if there are no more characters to be removed.

```python
def thirdCase(password, changes):
    i = 0
    while i < len(password) - 2:
        if password[i] == password[i + 1] == password[i + 2]:
            if changes['types'] != 0 and changes['removeChr'] == 0:
                changes['types'] = changes['types'] - 1
            if changes['removeChr'] != 0:
                changes['removeChr'] = changes['removeChr'] - 1
            if changes['removeChr'] == 0:
                i = i + 2

            changes['repeatingChr'] = changes['repeatingChr'] + 1
        i = i + 1
    return sum(list(changes.values()))
```