Let's review the conditions a password must meet in order to be safe:

1. It has at least 6 characters
2. It has at most 20 characters.
3. It must contain at least one lowercase letter, at least one uppercase letter, and at least one digit.
4. It must NOT contain three repeating characters in a row

Now, let us review which operations can solve each of the possible issues:

1 -> add more chars

2 -> delete chars

3 - > add chars, replace char

4 -> delete chars, add chars, replace char

To compute the best order to solve each problem (order that results in least changes), observe that we might solve problem 4 simply by solving problems 1, 2 or 3. Also, solving problem 1 could also solve problem 3.

Therefore, to get the least number of moves, we can treat the conditions in this order: 1, 2, 3, 4**.**

2,1,4,3 is just as good.

Before we start, let's analyze how exactly we can meet condition 4:

1. By replacing chars: replacing every third char in a substring of repeating characters is the best move:
   XXXXX - > XXaXX
2. By inserting chars: inserting a new char (one that does not exist in our password yet) is the best move:
   XXXX - > XXaXX
3. By deleting chars: best move is to delete from substrings of length that have a lower modulo 3 value:
   XXXXX (needs one more change in order to break substring) - > XXXX (needs one more change)
   XXXX (needs one more change) –> XXX (needs one more change)
   XXXXXX (needs two more changes) –> XXXXX (need one more change)

Therefore, we define **priority** as the lowest modulo 3 value for each substring length and memorize the substrings in a priority queue.

My solution follows the next algorithm:

1. Compute missing conditions
2. Add chars until **condition 1 is met**
   a. If there are missing categories of chars, add them
   b. If there exists substrings of repeating characters, break them up in order of **priority**
3. Delete chars until **condition 2 is met**
4. Replace chars until **condition 4 is met (*)**
   a. If there are missing categories of chars, add them
5. Replace chars until **condition 3 is met**