Kick Start 2022 - Round A

Analysis: Speed Typing

For better understanding, let us define some variables:
N: Length of string $\mathbf{I}$
M: Length of string $\mathbf{P}$

Test Set 1

All the characters in String $\mathbf{I}$ are the same, let us say that character is C. This means that $\mathbf{I}$ is made up entirely of character C occurring N times.
We need to check if $\mathbf{P}$ contains C at least N number of times (recall that N denotes the length of string $\mathbf{I}$). This can be done by simply maintaining a count of C in $\mathbf{P}$. If this count is greater than or equal to N, then the number of characters to be removed would be $(M - N)$, which is the number of extra characters in $\mathbf{P}$. Else, the answer is `IMPOSSIBLE`.

Time and Space Complexity: All operations including counting the appearances of C in $\mathbf{P}$ can be done in $O(M)$ time, which is the overall time complexity of this solution. Extra space required would be of the order of $O(1)$.

Test Set 2

Observe that we are trying to condense $\mathbf{P}$ to $\mathbf{I}$ by deleting some characters from $\mathbf{P}$ without disturbing their order. This directly leads us to the definition of a subsequence:
A subsequence is a sequence that can be derived from another sequence by deleting some elements without changing the order of the remaining elements.

- If $\mathbf{I}$ is not a subsequence of $\mathbf{P}$, the answer is `IMPOSSIBLE`. This can take place when:
  - Case 1: $\mathbf{P}$ is shorter than $\mathbf{I}$
  - Case 2: A character in $\mathbf{I}$ is not present in $\mathbf{P}$
  - Case 3: A character in $\mathbf{I}$ is present in $\mathbf{P}$, but not at the right place
- If $\mathbf{I}$ is a subsequence of $\mathbf{P}$, then $\mathbf{P}$ contains all characters of $\mathbf{I}$ in the same order along with some extra characters (possibly 0). Barbara will have to hit backspace on these extra characters to correct the string. The answer would be $(M - N)$.

Time and Space Complexity: Subsequence check can be done in $O(M)$ time using a two-pointer approach. The overall time complexity of this solution comes out to be $O(M)$ as well. Extra space required would be of the order of $O(1)$.

Pseudocode for Subsequence Check:

```
FUNCTION checkSubSequence(String I, String P) RETURNS BOOLEAN

    // Assign lengths of I and P to variables N and M
    N <- Length of I
    M <- Length of P

    // Declare pointers ptrI and ptrP to the current position in I and P respectively
    // and assign them to position 0
    ptrI <- 0
    ptrP <- 0

    // Traverse both strings, and compare current character of P with first unmatched char of I
    // While making sure the pointers do not go out of bounds of their respective strings
    WHILE ptrI is less than N AND ptrP is less than M

        // If characters at current positions match, then move ahead in both I and P
        IF I[ptrI] equals P[ptrP]
          ptrI <- ptrI + 1
          ptrP <- ptrP + 1

        // If the characters at current the positions do not match, then move ahead only in P
        ELSE
          ptrP <- ptrP + 1

        ENDIF
```

```
    ENDWHILE


    // If we reach past the end of I, we have successfully matched all the characters of I
    // to some characters of P
    // If not, some characters in I remain unmatched and it is not a subsequence of P
    IF ptrI equals N
      RETURN True;

    ELSE
      RETURN False;

    ENDIF

ENDFUNCTION
```