Exercício 1

Considere que a gente só possa fazer três operações com string: adicionar um caractere a ele, remover um caractere dele ou substituir um caractere por outro. Dizemos que uma string é 'one edit' de outra se ela for o resultado da original a partir de UMA SÓ dessas operações. Escreva um método que determine se uma string é 'one edit' de outra. RESPOSTA:

```
export const isOneEditString = (
    source: string,
    comparision: string
): boolean => {
  if (
    comparision.length > source.length + 1 ||
    comparision.length < source.length - 1
  ) {
    return false;
  }
  let communCharQuantity = 0;
  for (const char of comparision) {
    if (source.indexOf(char) !== -1) {
      communCharQuantity++;
    }
  }
  return (
    communCharQuantity <= source.length + 1 &&
    communCharQuantity >= source.length - 1
  );
};
```

Exercício 2

Implemente um método que performe uma compressão de string usando a contagem dos caracteres repetidos em sequência. Caso o resultado final tamanho maior do que a string inicial, seu programa deve retornar a string inicial RESPOSTA:

```
export const stringCompression = (input: string): string => {
  const substrings: string[] = [];
  let lastChar = input[0];
  let charCount = 0;
  for (const char of input) {
    if (char !== lastChar) {
      substrings.push(lastChar + charCount);
      lastChar = char;
      charCount = 0;
    }
    charCount++;
  }
  substrings.push(lastChar + charCount);
  let result = "";
```

```
  for (const key of substrings) {
    result += key;
  }
  return result.length > input.length ? input : result;
};
```