Caesar Cipher Encryptor

Input: string = "xyz"

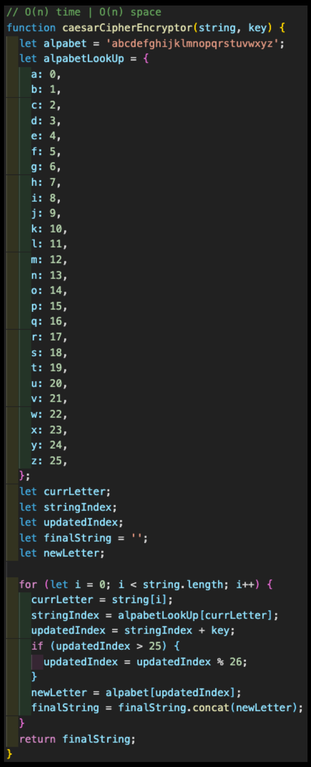
key = 2

Output: "zab"

Input: A non-empty string of lower case letters

A non-negative integer representing a key

Output: A new string obtained by shifting every letter in the input string by K positions in the alphabet where K is the key



```javascript
// O(n) time | O(n) space
function caesarCipherEncryptor(string, key) {
  let alpabet = 'abcdefghijklmnopqrstuvwxyz';
  let alpabetLookUp = {
    a: 0,
    b: 1,
    c: 2,
    d: 3,
    e: 4,
    f: 5,
    g: 6,
    h: 7,
    i: 8,
    j: 9,
    k: 10,
    l: 11,
    m: 12,
    n: 13,
    o: 14,
    p: 15,
    q: 16,
    r: 17,
    s: 18,
    t: 19,
    u: 20,
    v: 21,
    w: 22,
    x: 23,
    y: 24,
    z: 25,
  };
  let currLetter;
  let stringIndex;
  let updatedIndex;
  let finalString = '';
  let newLetter;

  for (let i = 0; i < string.length; i++) {
    currLetter = string[i];
    stringIndex = alpabetLookUp[currLetter];
    updatedIndex = stringIndex + key;
    if (updatedIndex > 25) {
      updatedIndex = updatedIndex % 26;
    }
    newLetter = alpabet[updatedIndex];
    finalString = finalString.concat(newLetter);
  }
  return finalString;
}
```

Time: O(n) (where n is the length of the input string) since we iterate over the entire string using a for loop

Space: O(n) (where n is the length of the input string) since we are concat the same # of letters as the input string.

} For the case where we get large Keys. We use the modulo operator (returns the remainder) and mod by 26 since 26 % 26 = 0 which is "a", 27 % 26 = 1 which is "b" and so on