1) A Fibonacci string is a precedence of the Fibonacci series. It works with any two characters of the English alphabet (as opposed to the numbers 0 and 1 in the Fibonacci series) as the initial items and concatenates them together as it progresses similarly to the Fibonacci series.

Examples

fibStr(3, ["j", "h"]) ⟶ "j, h, hj"

fibStr(5, ["e", "a"]) ⟶ "e, a, ae, aea, aeaae"

fibStr(6, ["n", "k"]) ⟶ "n, k, kn, knk, knkkn, knkknknk"

Notes

All values for n will be at least 2.

2) Create a function that takes numbers as arguments, adds them together, and returns the product of digits until the answer is only 1 digit long.

Examples

sumDigProd(16, 28) ⟶ 6

// 16 + 28 = 44

// 4 * 4 = 16

// 1 * 6 = 6

sumDigProd(0) ⟶ 0

sumDigProd(1, 2, 3, 4, 5, 6) ⟶ 2

Notes

The input of the function is at least one number.

3) Create a function that decomposes an address string into an array of five substrings:

Street Number

Street Name

City Name

State

Zip Code

Examples

decomposeAddress("557 Farmer Rd Corner, MT 59105")

→ ["557", "Farmer Rd", "Corner", "MT", "59105"]

decomposeAddress("3315 Vanity St Beverly Hills, CA 90210")

→ ["3315", "Vanity St", "Beverly Hills", "CA", "90210"]

decomposeAddress("8919 Scarecrow Ct Idaho Falls, ID 80193")

→ ["8919", "Scarecrow Ct", "Idaho Falls", "ID", "80193"]

Notes

All street extensions will be shortened to two-letter formats.

4)Write a function that returns the amount of possible combinations when drawing the given amount of cards from a deck of cards.

The function must take two inputs: k is the amount of cards to draw. n the total amount of cards in the deck.

For example, a poker hand can be described as a 5-combination (k = 5) of cards from a 52 card deck (n = 52). The 5 cards of the hand are all distinct, and the order of cards in the hand does not matter. There are 2,598,960 such combinations.

The amount of combinations should be returned as an integer.

Examples

combinations(52, 52) → 1

combinations(5, 52) → 2598960

combinations(10, 52) → 15820024220

Notes

Try solving this natively without any imports.

Remember to return result as integer.