1. Write four functions that directly mutate a list:

   1. repeat(lst, n): Repeat lst n times.
   2. add(lst, x): Adds x to the end of the lst.
   3. remove(lst, m, n): Removes all elements between indices m and n inclusive in lst.
   4. concat(lst, x): concatenates lst with x (another list).

Examples

lst = [1, 2, 3, 4]

repeat(lst, 3) [1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4]

add(lst, 1) [1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1]

remove(lst, 1, 12) [1]

concat(lst, [3, 4]) [1, 3, 4]

2. The classic game of Mastermind is played on a tray on which the Mastermind conceals a code and the Guesser has 10 tries to guess it. The code is a sequence of 4 (or 6, sometimes more) pegs of different colors. Each guess is a corresponding sequence of 4 (or more) pegs of different colors. A guess is "correct" when the color of every peg in the guess exactly matches the corresponding peg in the Mastermind's code.

After each guess by the Guesser, the Mastermind will give a score comprising black & white pegs, not arranged in any order:

- Black peg == guess peg matches the color of a code peg in the same position.
- White peg == guess peg matches the color of a code peg in another position.

Create a function that takes two strings, code and guess as arguments, and returns the score in a dictionary.

- The code and guess are strings of numeric digits
- The color of the pegs are represented by numeric digits
- no "peg" may be double-scored

Examples

guess_score("1423", "5678") {"black": 0, "white": 0}

guess_score("1423", "2222") {"black": 1, "white": 0}

guess_score("1423", "1234") {"black": 1, "white": 3}

guess_score("1423", "2211") {"black": 0, "white": 2}

3. Create a function that takes a list lst and a number N and returns a list of two integers from lst whose product equals N.

Examples

two_product([1, 2, -1, 4, 5], 20) [4, 5]

two_product([1, 2, 3, 4, 5], 10) [2, 5]

two_product([100, 12, 4, 1, 2], 15) None

4. In this challenge, sort a list containing a series of dates given as strings. Each date is given in the format DD-MM-YYYY_HH:MM:

"12-02-2012_13:44"

The priority of criteria used for sorting will be:

- Year
- Month
- Day
- Hours
- Minutes

Given a list lst and a string mode, implement a function that returns:

- if mode is equal to "ASC", the list lst sorted in ascending order.
- if mode is equal to "DSC", the list lst sorted in descending order.

Examples

sort_dates(["10-02-2018_12:30", "10-02-2016_12:30", "10-02-2018_12:15"], "ASC") ["10-02-2016_12:30", "10-02-2018_12:15", "10-02-2018_12:30"]

sort_dates(["10-02-2018_12:30", "10-02-2016_12:30", "10-02-2018_12:15"], "DSC") ["10-02-2018_12:30", "10-02-2018_12:15", "10-02-2016_12:30"]

sort_dates(["09-02-2000_10:03", "10-02-2000_18:29", "01-01-1999_00:55"], "ASC") ["01-01-1999_00:55", "09-02-2000_10:03", "10-02-2000_18:29"]
5. Write a function that selects all words that have all the same vowels (in any order and/or number) as the first word, including the first word.

Examples

same_vowel_group(["toe", "ocelot", "maniac"]) ["toe", "ocelot"]

same_vowel_group(["many", "carriage", "emit", "apricot", "animal"]) ["many"]

same_vowel_group(["hoops", "chuff", "bot", "bottom"]) ["hoops", "bot", "bottom"]

6. Create a function that takes a list of more than three numbers and returns the Least Common Multiple (LCM).

Examples

lcm_of_list([1, 2, 3, 4, 5, 6, 7, 8, 9, 10]) 2520

lcm_of_list([13, 6, 17, 18, 19, 20, 37]) 27965340

lcm_of_list([44, 64, 12, 17, 65]) 2333760