**1) Product of other numbers**

Write a function 1that takes a list of integers and returns a list of the products.

For example, given:

[1, 7, 3, 4]

your function would return:

[84, 12, 28, 21]

by calculating:

[7\*3\*4, 1\*3\*4, 1\*7\*4, 1\*7\*3]

**Do not use division in your solution.**

**2) Contense meeting times**

Your company built an in-house calendar tool called HiCal. You want to add a feature to see the times in a day when everyone is available. To do this, you’ll need to know when any team is having a meeting. In HiCal, a meeting is stored as tuples ↴ of integers (start\_time, end\_time). These integers represent the number of 30-minute blocks past 9:00am.

For example:

(2, 3) # meeting from 10:00 – 10:30 am

(6, 9) # meeting from 12:00 – 1:30 pm

Write a function condense\_meeting\_times() that takes a list of meeting time ranges and returns a list of condensed ranges.

For example, given:

[(0, 1), (3, 5), (4, 8), (10, 12), (9, 10)]

your function would return:

[(0, 1), (3, 8), (9, 12)]

Do not assume the meetings are in order. The meeting times are coming from multiple teams.

Write a solution that's efficient even when we can't put a nice upper bound on the numbers representing our time ranges. Here we've simplified our times down to the number of 30-minute slots past 9:00 am. But we want the function to work even for very large numbers, like Unix timestamps. In any case, the spirit of the challenge is to merge meetings where start\_time and end\_time don't have an upper bound.

3) Suppose we could access yesterday's stock prices as a list, where:

The indices are the time in minutes past trade opening time, which was 9:30am local time.

The values are the price in dollars of Apple stock at that time.

So if the stock cost $500 at 10:30am, stock\_prices\_yesterday[60] = 500.

Write an efficient function that takes stock\_prices\_yesterday and returns the best profit

I could have made from 1 purchase and 1 sale of 1 Apple stock yesterday.

Input sample stock\_prices\_yesterday = [10,7,5,8,11,9]

4) Implement a queue with 2 stacks. Your queue should have an enqueue and a dequeue method and it should be "first in first out" (FIFO).

Optimize for the time cost of mm calls on your queue. These can be any mix of enqueue and dequeue calls.

Assume you already have a stack implementation and it gives O(1)O(1) time push and pop.

5) Your company delivers breakfast via autonomous quadcopter drones. And something mysterious has happened.

Each breakfast delivery is assigned a unique ID, a positive integer. When one of the company's 100 drones takes off with a delivery, the delivery's ID is added to a list, delivery\_id\_confirmations. When the drone comes back and lands, the ID is again added to the same list.

After breakfast this morning there were only 99 drones on the tarmac. One of the drones never made it back from a delivery. We suspect a secret agent from Amazon placed an order and stole one of our patented drones. To track them down, we need to find their delivery ID.

Given the list of IDs, which contains many duplicate integers and one unique integer, find the unique integer.

The IDs are not guaranteed to be sorted or sequential. Orders aren't always fulfilled in the order they were received, and some deliveries get cancelled before takeoff

6) **You're working on a secret team solving coded transmissions.**

Your team is scrambling to decipher a recent message, worried it's a plot to break into a major European National Cake Vault. The message has been *mostly* deciphered, but all the words are backward! Your colleagues have handed off the last step to you.

Write a function reverse\_words() that takes a message as a list of characters and reverses the order of the words in place. ↴

Why a list of characters instead of a string?

The goal of this question is to practice manipulating strings *in place*. Since we're modifying the message, we need a **mutable ↴**type like a list, instead of Python's *immutable* strings.

For example:

message = [ 'c', 'a', 'k', 'e', ' ',

'p', 'o', 'u', 'n', 'd', ' ',

's', 't', 'e', 'a', 'l' ]

reverse\_words(message)

# Prints: 'steal pound cake'

print ''.join(message)

When writing your function, assume the message contains only letters and spaces, and all words are separated by one space.

7) Is the word an anagram of a palindrome?

A palindrome is a word that reads the same forward and backwards (eg, "racecar", "tacocat"). An anagram is a rescrambling of a word (eg for "racecar", you could rescramble this as "arceace").

Determine if the given word is a re-scrambling of a palindrome.

The word will only contain lowercase letters, a-z

8) Write a solution with O(n) time complexity and O(1) additional space complexity, since this is what you would be asked to do during a real interview.

Given an array a that contains only numbers in the range from 1 to a.length, find the first duplicate number for which the second occurrence has the minimal index. In other words, if there are more than 1 duplicated numbers, return the number for which the second occurrence has a smaller index than the second occurrence of the other number does. If there are no such elements, return -1.

Example

For a = [2, 3, 3, 1, 5, 2], the output should be

firstDuplicate(a) = 3.

There are 2 duplicates: numbers 2 and 3. The second occurrence of 3 has a smaller index than than second occurrence of 2 does, so the answer is 3.

For a = [2, 4, 3, 5, 1], the output should be

firstDuplicate(a) = -1.

Input/Output

[time limit] 4000ms (py)

[input] array.integer a

Guaranteed constraints:

1 <= a.length <= 105,

1 <= a[i] <= a.length.

[output] integer

The element in a that occurs in the array more than once and has the minimal index for its second occurrence. If there are no such elements, return -1.