Codewriting

You are given two arrays of positive integers, `firstArray` and `secondArray`, as well as an integer `target`. Your task is to find the number of pairs of non-empty sub-arrays — one from `firstArray` and one from `secondArray` — such that the total sum of numbers in them is equal to `target`.

Formally, your task is to find the number of tuples of integers `(i, j, n, m)`, where

- `1 <= n < firstArray.length`,
- `0 <= i < firstArray.length - n`,
- `1 <= m < secondArray.length`,
- `0 <= j < secondArray.length - m`,

such that `firstArray[i] + firstArray[i + 1] + ... + firstArray[i + n] + secondArray[j] + secondArray[j + 1] + ... + secondArray[j + m] = target`.

Example

For `firstArray = [5, 2, 1, 6, 4]`, `secondArray = [3, 5]`, and `target = 10`, the output should be `solution(firstArray, secondArray, target) = 4`.

Codewriting

You are given two arrays of positive integers, `firstArray` and `secondArray`, as well as an integer `target`. Your task is to find the number of pairs of non-empty sub-arrays — one from `firstArray` and one from `secondArray` — such that the total sum of numbers in them is equal to `target`.

Formally, your task is to find the number of tuples of integers `(i, j, n, m)`, where

- `1 <= n < firstArray.length`,
- `0 <= i < firstArray.length - n`,
- `1 <= m < secondArray.length`,
- `0 <= j < secondArray.length - m`,

such that `firstArray[i] + firstArray[i + 1] + ... + firstArray[i + n] + secondArray[j] + secondArray[j + 1] + ... + secondArray[j + m] = target`.

Example

For `firstArray = [5, 2, 1, 6, 4]`, `secondArray = [3, 5]`, and `target = 10`, the output should be `solution(firstArray, secondArray, target) = 4`.

DESC

HISTORY

RULES

README

SETTINGS

MORE

Codewriting

Given an array of integers `numbers`, distribute all of its integers between two arrays, `first` and `second`, based on the following rules:

- The first number, `numbers[0]`, goes to the `first` array.
- The second number, `numbers[1]`, goes to the `second` array.
- Each following number, `numbers[i]` where `i > 1`, goes to the array with the higher number of elements that are **strictly greater** than the `numbers[i]`. In case of a tie, `numbers[i]` goes to the array with a lower length. If it is still a tie, `numbers[i]` goes to the `first` array.

Your task is to return a single array - the combination of `first` and `second` by appending all elements of `second` to the end of `first`.

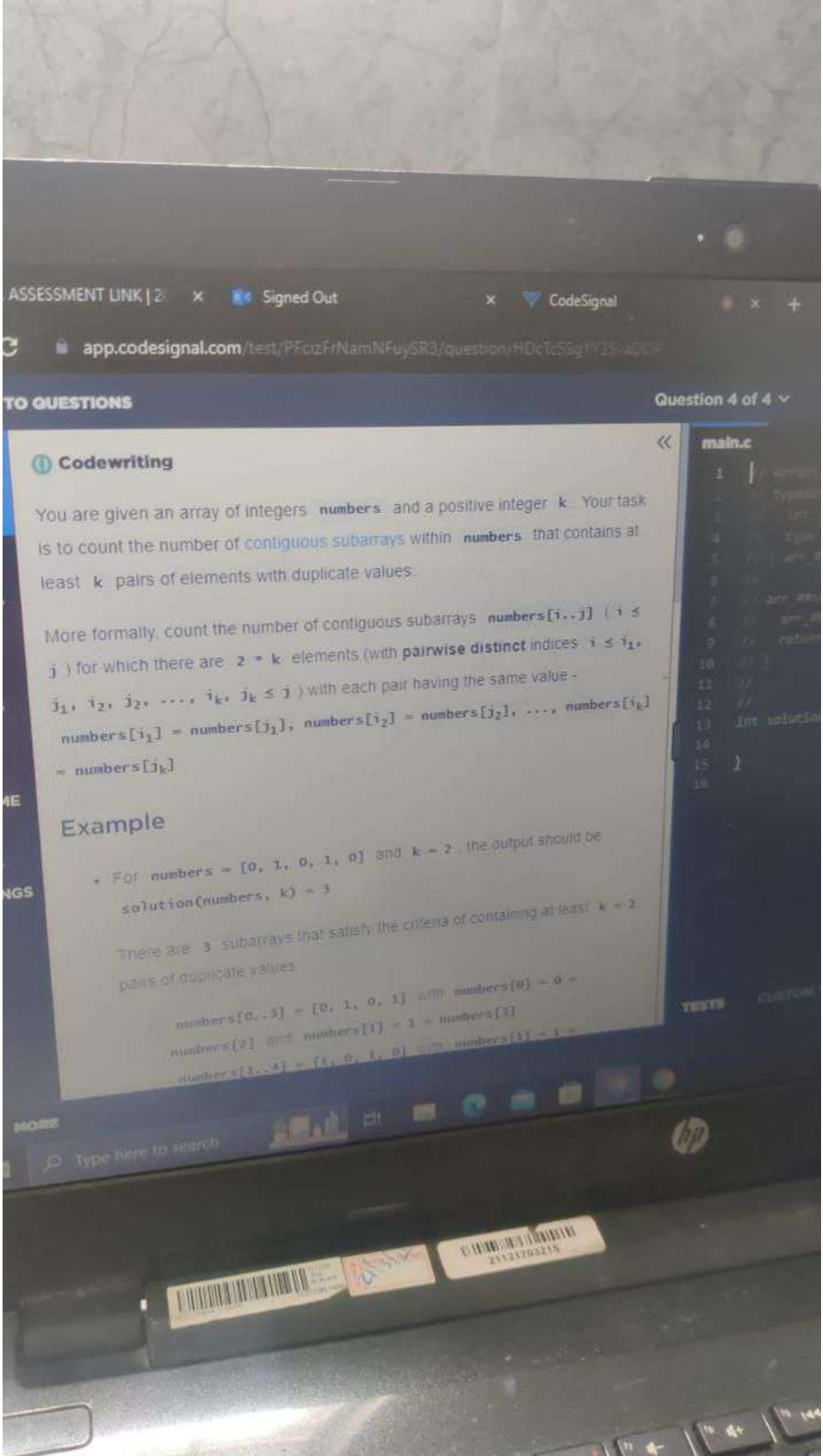
Note: You are not expected to provide the most optimal solution, but a solution with time complexity not worse than $O(numbers.length^2)$ *will fit within the execution time limit.*

main.java

```
1  int[]
2
3  }
4
```

TESTS

Search


ASSESSMENT LINK | 2
Signed Out
CodeSignal
app.codesignal.com/test/PFozFrNamNFuySR3/question/HDcTcSSgYY3SvaZDP
TO QUESTIONS
Question 4 of 4
Codewriting
You are given an array of integers numbers and a positive integer k. Your task is to count the number of contiguous subarrays within numbers that contains at least k pairs of elements with duplicate values.
More formally, count the number of contiguous subarrays numbers[i..j] (i ≤ j) for which there are 2 * k elements (with pairwise distinct indices i ≤ i1, j1, i2, j2, ..., ik, jk ≤ j) with each pair having the same value - numbers[i1] = numbers[j1], numbers[i2] = numbers[j2], ..., numbers[ik] = numbers[jk].
Example
For numbers = [0, 1, 0, 1, 0] and k = 2, the output should be solution(numbers, k) = 3.
There are 3 subarrays that satisfy the criteria of containing at least k = 2 pairs of duplicate values:
numbers[0..3] = [0, 1, 0, 1] with numbers[0] = 0 = numbers[2] and numbers[1] = 1 = numbers[3]
numbers[1..4] = [1, 0, 1, 0] with numbers[1] = 1 =
main.c
int solution
TESTS
Type here to search

QUESTIONS

- **[input] array.integer a**

  An array of positive integers.

  Guaranteed constraints:

  $1 \leq a.length \leq 10^5$,

  $1 \leq a[i] \leq 10^9$.

- **[output] integer64**

  The number of cyclic pairs between `i` and `j`, such that `a[i]` and `a[j]` has the same number of digits and `a[i]` is equal to a cyclic shift of `a[j]`.

[Java] Syntax Tips

```
// Prints help message to the console
// Returns a string
//
// Globals declared here will cause a compilatic
// declare variables inside the function instead
String helloWorld(String name) {
    System.out.println("This prints to the consc
    return "Hello, " + name;
}
```

1 Codewriting

A *cyclic shift* is the operation of rearranging the digits in a number (in decimal format) by moving some digits at the end of the number to before the beginning of the number, while shifting all other digits to the next position.

Given two integers of the same length `a` and `b`, `a` would be a cyclic pair of `b` if it's possible for `a` to become equal to `b` after performing cyclic shifts on `a` - moving 0 or more ending digits to the beginning while shifting all other digits to the next position in the same order.

Given an array of positive integers `a`, your task is to count the number of cyclic pairs `i` and `j` (where `0 ≤ i < j < a.length`), such that `a[i]` and `a[j]` have the same number of digits and `a[i]` is equal to a cyclic shift of `a[j]`

① Codewriting

A *cyclic shift* is the operation of rearranging the digits in a number (in decimal format) by moving some digits at the end of the number to before the beginning of the number, while shifting all other digits to the next position.

Given two integers of the same length `a` and `b`, `a` would be a cyclic pair of `b` if it's possible for `a` to become equal to `b` after performing cyclic shifts on `a` - moving 0 or more ending digits to the beginning while shifting all other digits to the next position in the same order.

Given an array of positive integers `a`, your task is to count the number of cyclic pairs `i` and `j` (where `0 ≤ i < j < a.length`), such that `a[i]` and `a[j]` have the same number of digits and `a[i]` is equal to a cyclic shift of `a[j]`

Given an array of positive integers `a`, your task is to count the number of cyclic pairs `i` and `j` (where `0 ≤ i < j < a.length`), such that `a[i]` and `a[j]` have the same number of digits and `a[i]` is equal to a cyclic shift of `a[j]`.

Example

For `a = [13, 5604, 31, 2, 13, 4560, 546, 654, 456]`, the output should be `solution(a) = 5`.

There are 5 cyclic pairs of numbers - pairs which are equal to each other after cyclic shifts.

- `a[0] = 13` and `a[2] = 31` (`i = 0` and `j = 2`).
- `a[0] = 13` and `a[4] = 13` (`i = 0` and `j = 4`).
- `a[2] = 31` and `a[4] = 13` (`i = 2` and `j = 4`).

4).

- a[1] = 5604 and a[5] = 4560 (i = 1 and j = 5).
- a[6] = 546 and a[7] = 654 (i = 6 and j = 7)

Note that a[6] = 546 and a[8] = 456 are not cyclic pairs — 546 can only be paired with cyclic shift of 546 , 465 and 654 .

Also, note that a[5] = 4560 and a[8] = 456 are not cyclic pairs because they have different number of digits.

Input/Output

- [execution time limit] 3 seconds (java)
- [memory limit] 1 GB
- [input] array.integer a

  An array of positive integers.

QUESTIONS

- **[input] array.integer a**

  An array of positive integers.

  Guaranteed constraints:

  $1 \leq a.length \leq 10^5$,

  $1 \leq a[i] \leq 10^9$.

- **[output] integer64**

  The number of cyclic pairs between `i` and `j`, such that `a[i]` and `a[j]` has the same number of digits and `a[i]` is equal to a cyclic shift of `a[j]`.

[Java] Syntax Tips

```
// Prints help message to the console
// Returns a string
//
// Globals declared here will cause a compilatic
// declare variables inside the function instead
String helloWorld(String name) {
    System.out.println("This prints to the consc
    return "Hello, " + name;
}
```

Given an array of positive integers `a`, your task is to count the number of cyclic pairs `i` and `j` (where `0 ≤ i < j < a.length`), such that `a[i]` and `a[j]` have the same number of digits and `a[i]` is equal to a cyclic shift of `a[j]`.

Example

For `a = [13, 5604, 31, 2, 13, 4560, 546, 654, 456]`, the output should be `solution(a) = 5`.

There are 5 cyclic pairs of numbers - pairs which are equal to each other after cyclic shifts.

- `a[0] = 13` and `a[2] = 31` (`i = 0` and `j = 2`).
- `a[0] = 13` and `a[4] = 13` (`i = 0` and `j = 4`).
- `a[2] = 31` and `a[4] = 13` (`i = 2` and `j = 4`).

4).

- a[1] = 5604 and a[5] = 4560 (i = 1 and j = 5).
- a[6] = 546 and a[7] = 654 (i = 6 and j = 7)

Note that a[6] = 546 and a[8] = 456 are not cyclic pairs — 546 can only be paired with cyclic shift of 546 , 465 and 654 .

Also, note that a[5] = 4560 and a[8] = 456 are not cyclic pairs because they have different number of digits.

Input/Output

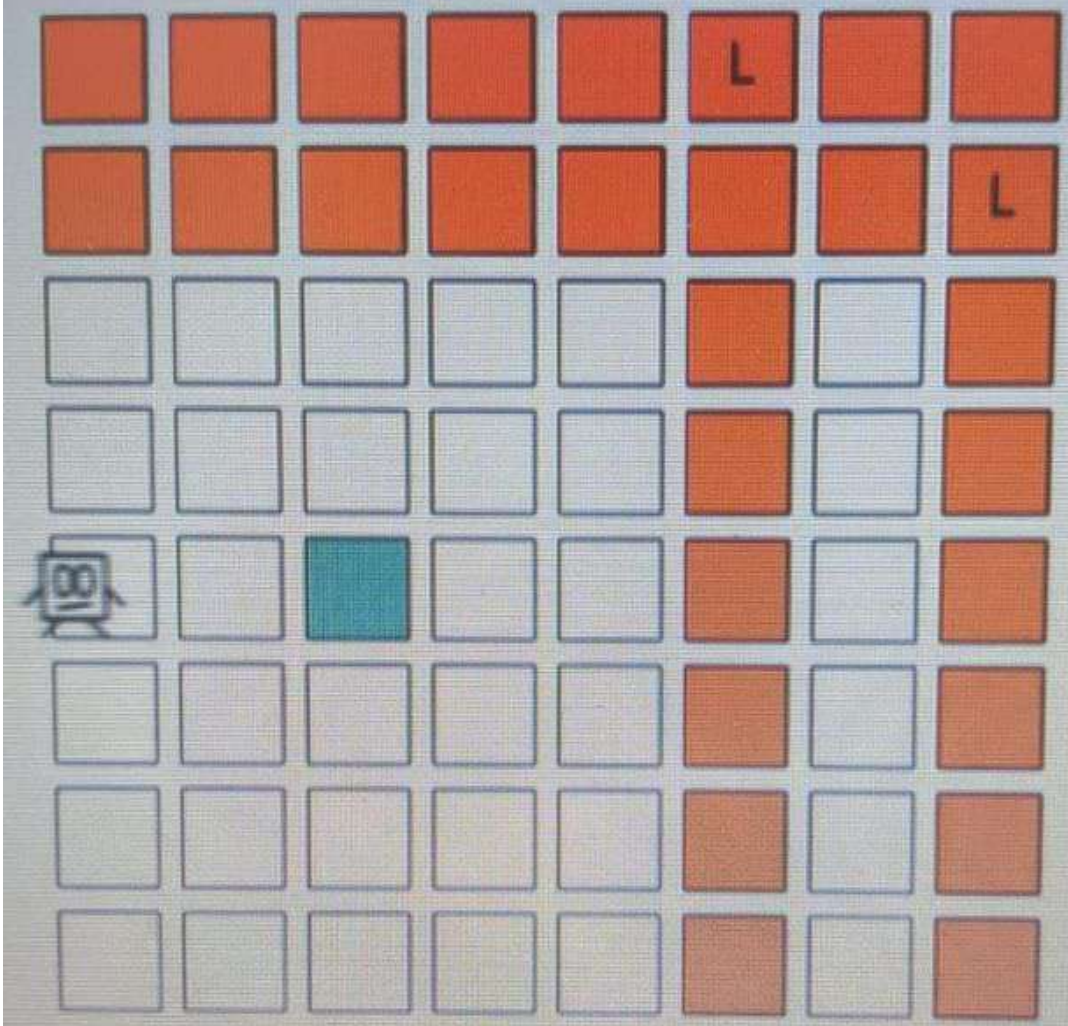
- [execution time limit] 3 seconds (java)
- [memory limit] 1 GB
- [input] array.integer a

  An array of positive integers

app.codesignal.com

ESTIONS

Expand to see the example video.



Current direction: Left

Current answer: 2

Answer: 2

Note: If you are unable to view the video properly, please use

pe here to search

Codewriting

Imagine a board of size `numRows x numColumns` with some lasers placed on it. These lasers are placed at coordinates specified in the two-dimensional array `laserCoordinates`, where `laserCoordinates[i]` is a two-element array containing coordinates for the center of the i^{th} laser. Lasers with a center in a cell `(row, column)` destroy everything in the *same row* (i.e. rows with index `row`) and the *same column* (i.e. columns with index `column`).

Now imagine there is a robot at coordinates `(curRow, curColumn)`. The robot can only move in a straight line, either *left*, *right*, *up*, or *down* within this board. Your task is to count the maximum number of cells that the robot can safely move through (in any direction) before being destroyed by lasers.

Note: You can assume that the initial cell is protected, and lasers cannot destroy the robot there even if they cover this cell in their destruction area.

Note: You are not expected to provide the most optimal

E ASSESSMENT LINK | : × S gned Out × C

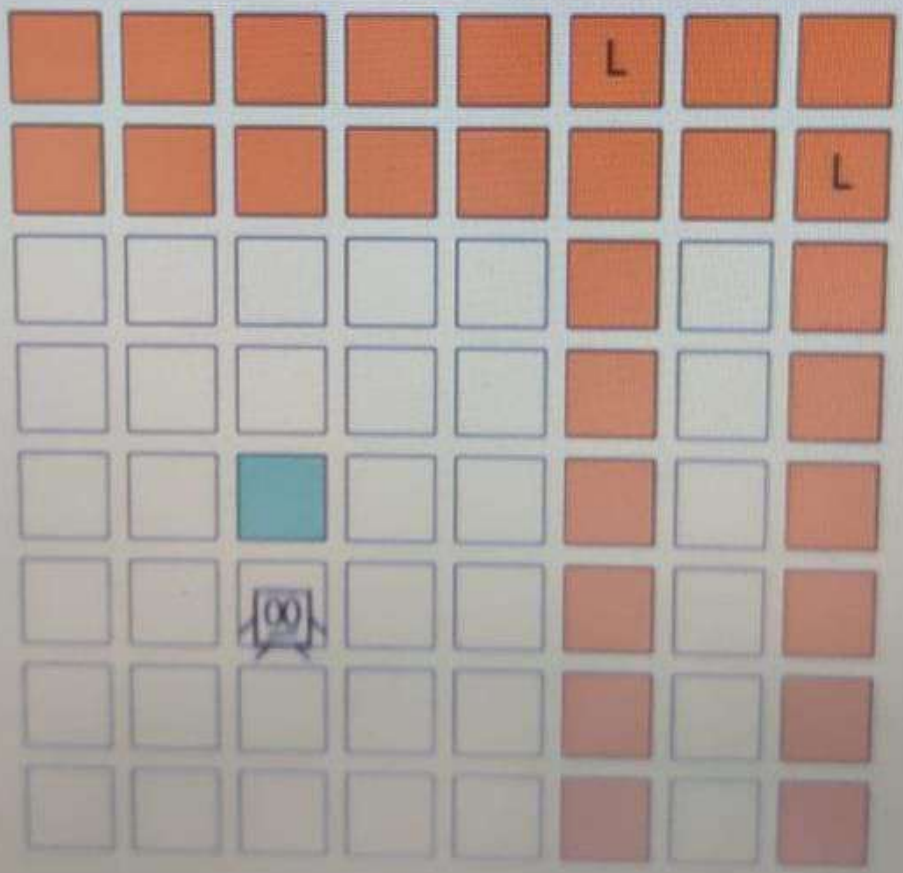
app.codesignal.com

TO QUESTIONS

Explanation:

Given the `8 × 8` board, there are two lasers with centers at cells `(1, 6)` and `(2, 8)`. The animation below shows that the longest safe path for the robot is `3`.

▼ Expand to see the example video.



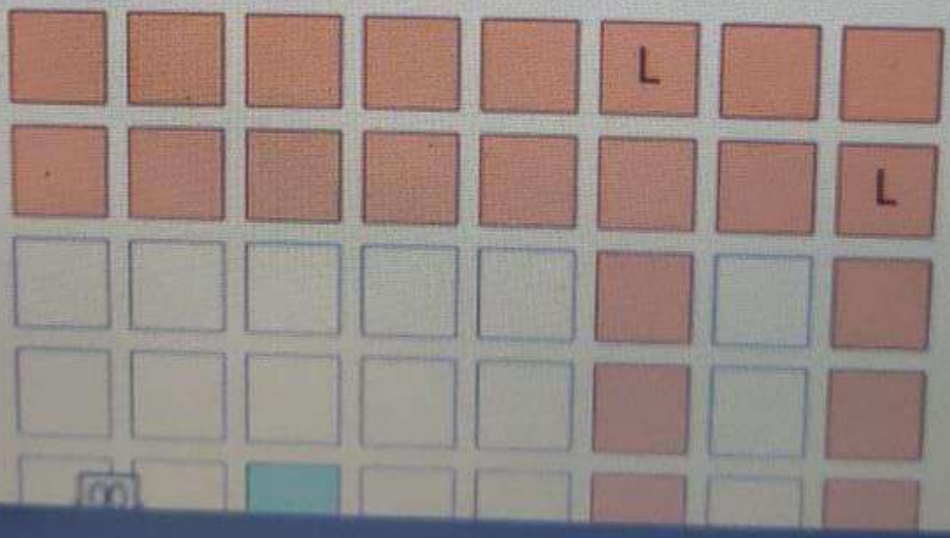
Example

For `numRows = 8`, `numColumns = 8`, `curRow = 5`, `curColumn = 3`, and `laserCoordinates = [[1, 6], [2, 8]]`, the output should be `solution(numRows, numColumns, curRow, curColumn, laserCoordinates) = 3`.

Explanation:

Given the `8 x 8` board, there are two lasers with centers at cells `(1, 6)` and `(2, 8)`. The animation below shows that the longest safe path for the robot is `3`.

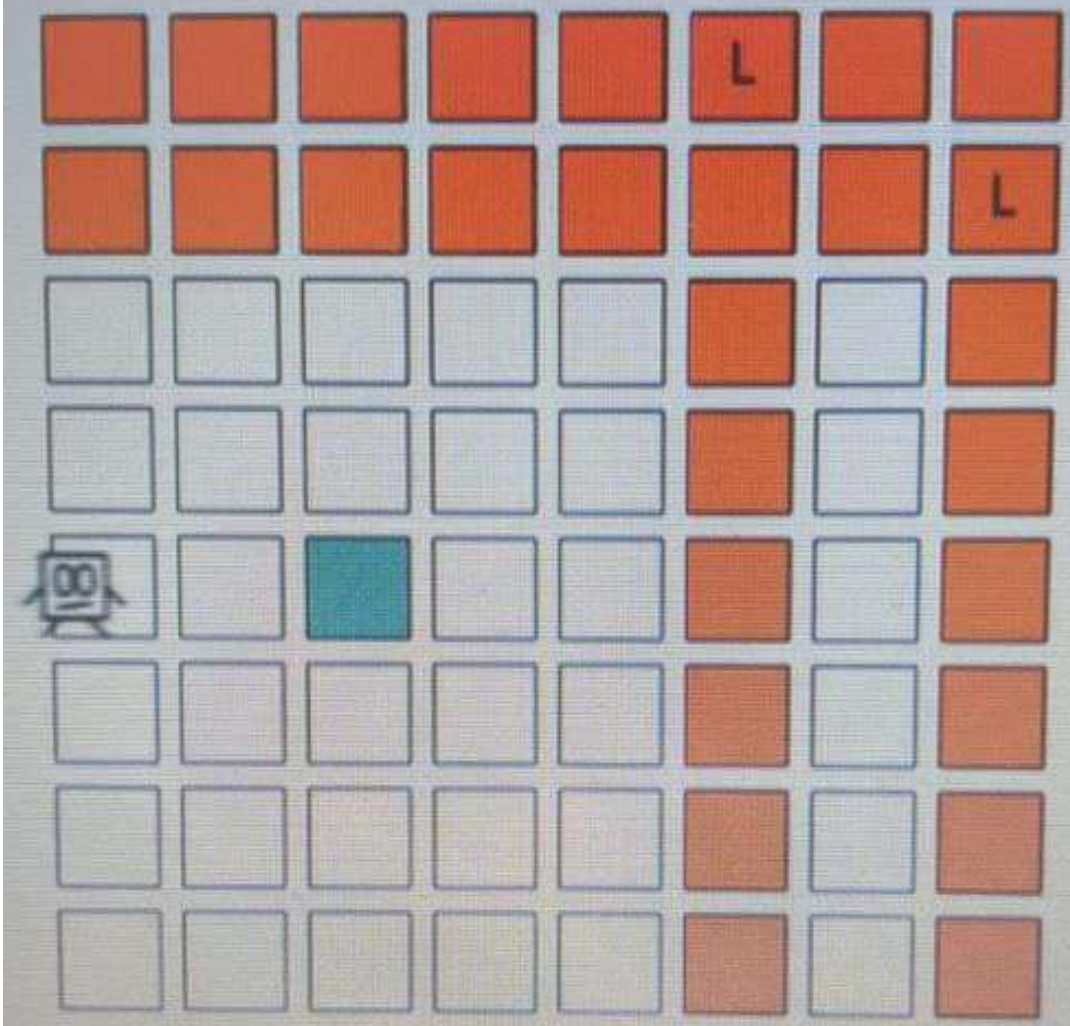
▼ Expand to see the example video.



app.codesignal.com

ESTIONS

Expand to see the example video.



Current direction: Left

Current answer: 2

Answer: 2

Note: If you are unable to view the video properly, please use

pe here to search

E ASSESSMENT LINK | : × Signed Out × C

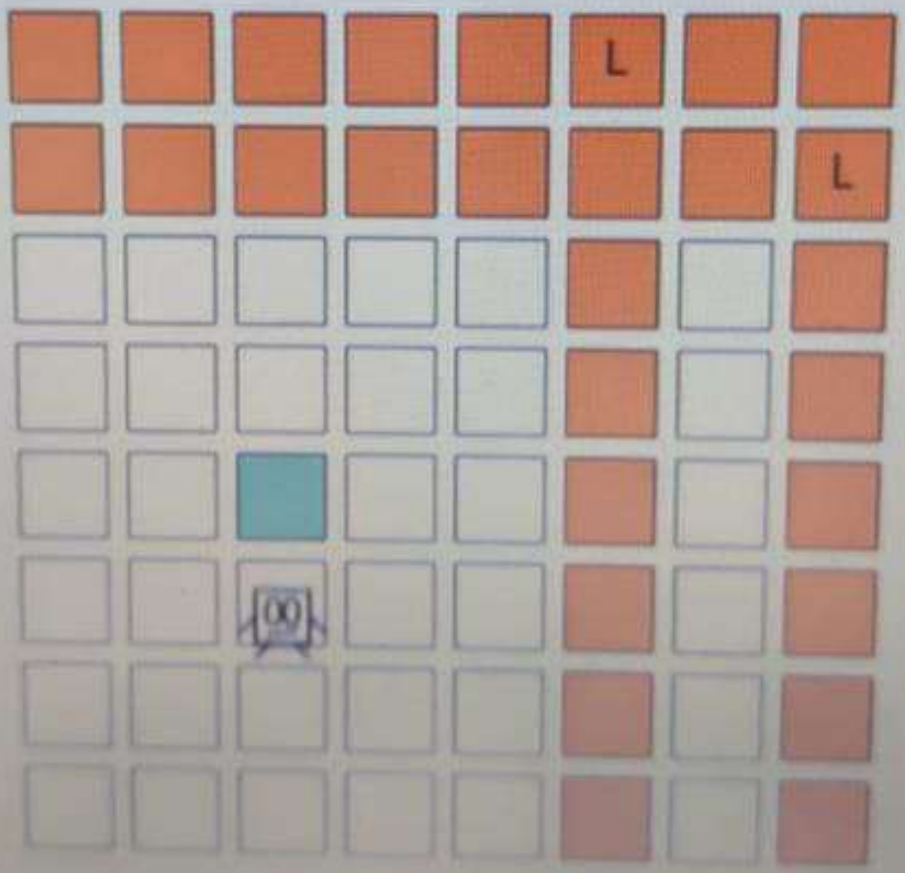
app.codesignal.com

TO QUESTIONS

Explanation:

Given the `8 x 8` board, there are two lasers with centers at cells `(1, 6)` and `(2, 8)`. The animation below shows that the longest safe path for the robot is `3`.

▼ Expand to see the example video.



Codewriting

Imagine a board of size `numRows x numColumns` with some lasers placed on it. These lasers are placed at coordinates specified in the two-dimensional array `laserCoordinates`, where `laserCoordinates[i]` is a two-element array containing coordinates for the center of the i^{th} laser. Lasers with a center in a cell `(row, column)` destroy everything in the *same row* (i.e. rows with index `row`) and the *same column* (i.e. columns with index `column`).

Now imagine there is a robot at coordinates `(curRow, curColumn)`. The robot can only move in a straight line, either *left*, *right*, *up*, or *down* within this board. Your task is to count the maximum number of cells that the robot can safely move through (in any direction) before being destroyed by lasers.

Note: You can assume that the initial cell is protected, and lasers cannot destroy the robot there even if they cover this cell in their destruction area.

Note: You are not expected to provide the most optimal

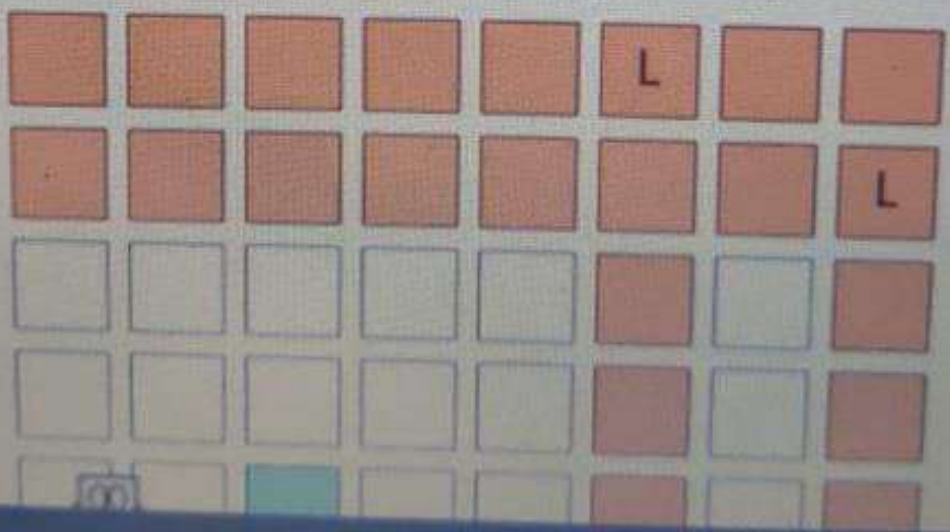
Example

For `numRows = 8`, `numColumns = 8`, `curRow = 5`, `curColumn = 3`, and `laserCoordinates = [[1, 6], [2, 8]]`, the output should be `solution(numRows, numColumns, curRow, curColumn, laserCoordinates) = 3`.

Explanation:

Given the `8 × 8` board, there are two lasers with centers at cells `(1, 6)` and `(2, 8)`. The animation below shows that the longest safe path for the robot is `3`.

▼ Expand to see the example video.



Codewriting

You are given an array of integers `memory` consisting of `0` s and `1` s which indicates whether the corresponding memory unit is free or not. `memory[i] = 0` means that the i^{th} memory unit is free, and `memory[i] = 1` means it's occupied.

The memory is aligned with segments of `8` units so all occupied memory blocks must start at an index divisible by `8` (e.g. `0` , `8` , `16` , etc).

Your task is to perform two types of queries:

- `alloc x` : Find the left-most **aligned** memory block of `x` consecutive free memory units and mark these units as occupied (ie: find the left-most contiguous subarray of `0` s, starting at the position `start` which is divisible by `8` , and replace all these memory units with `1` s).
    - If there is no proper aligned memory block with `x` consecutive free units, return `-1` ; otherwise return the index of the first position of the allocated block segment and assign an `ID` to every

single element in the block, based on an `atomic counter` *(the counter starts at 1 and is incremented on every successful alloc operation)*.

- Note: `X` may be greater than `8`, so the block may cover more than one memory segment.

- `erase ID` : If there exists an allocated memory block with element ids equal to `ID`, free all its memory units (set all of the bits in the block to `0`).
  - Return the length of the deleted memory block. If there is no such `ID` or the block with this `ID` has already been deleted, return `-1`.

The queries are given in the following format:

- `queries` is an array of 2-elements arrays;
- if `queries[i][0] = 0` then this is an `alloc` type query, where `X = queries[i][1]`;
- if `queries[i][0] = 1` then this is an `erase` type query, where `ID = queries[i][1]`.

Return an array containing the results of all the

```
1  int[] solution(i
2
3  }
4
```

TESTS CUSTOM TEST

app.codesigna

Search

single element in the block, based on an `atomic counter` *(the counter starts at 1 and is incremented on every successful alloc operation)*.

- Note: `X` may be greater than `8`, so the block may cover more than one memory segment.

- `erase ID`: If there exists an allocated memory block with element ids equal to `ID`, free all its memory units (set all of the bits in the block to `0`).
  - Return the length of the deleted memory block. If there is no such `ID` or the block with this `ID` has already been deleted, return `-1`.

The queries are given in the following format:

- `queries` is an array of 2-elements arrays;
- if `queries[i][0] = 0` then this is an `alloc` type query, where `X = queries[i][1]`;
- if `queries[i][0] = 1` then this is an `erase` type query, where `ID = queries[i][1]`.

Return an array containing the results of all the

```
1  int[] solution(i
2
3  }
4
```

TESTS CUSTOM TEST

app.codesigna

Search

Codewriting

You are given an array of integers `memory` consisting of `0` s and `1` s which indicates whether the corresponding memory unit is free or not. `memory[i] = 0` means that the i^{th} memory unit is free, and `memory[i] = 1` means it's occupied.

The memory is aligned with segments of `8` units so all occupied memory blocks must start at an index divisible by `8` (e.g. `0`, `8`, `16`, etc).

Your task is to perform two types of queries:

- `alloc x` : Find the left-most **aligned** memory block of `x` consecutive free memory units and mark these units as occupied (ie: find the left-most contiguous subarray of `0` s, starting at the position `start` which is divisible by `8`, and replace all these memory units with `1` s).
    - If there is no proper aligned memory block with `x` consecutive free units, return `-1`; otherwise return the index of the first position of the allocated block segment and assign an `ID` to every

DESC

HISTORY

RULES

README

SETTINGS

MORE

Codewriting

Given an array of integers `numbers`, distribute all of its integers between two arrays, `first` and `second`, based on the following rules:

- The first number, `numbers[0]`, goes to the `first` array.
- The second number, `numbers[1]`, goes to the `second` array.
- Each following number, `numbers[i]` where `i > 1`, goes to the array with the higher number of elements that are **strictly greater** than the `numbers[i]`. In case of a tie, `numbers[i]` goes to the array with a lower length. If it is still a tie, `numbers[i]` goes to the `first` array.

Your task is to return a single array - the combination of `first` and `second` by appending all elements of `second` to the end of `first`.

Note: You are not expected to provide the most optimal solution, but a solution with time complexity not worse than $O(numbers.length^2)$ *will fit within the execution time limit.*

main.java

```
1  int[]
2
3  }
4
```

TESTS

Search

DESC

HISTORY

RULES

README

SETTINGS

Codewriting

Given an array of integers `numbers`, your task is to count all numbers that are **not** equal to `numbers[0]` or `numbers[1]`, if they exist in the array.

Assume that array indices are 0-based.

Note: You are not expected to provide the most optimal solution, but a solution with time complexity not worse than $O(numbers.length^2)$ *will fit within the execution time limit.*

Example

- For `numbers = [4, 3, 2, 3, 2, 5, 4, 3]`, the output should be `solution(numbers) = 3`.

  Explanation:

  `numbers[0]` and `numbers[1]` are `4` and `3` respectively. There are three elements in `numbers` that are not equal to `4` or `3`:

  - `numbers[2] = 2`
  - `numbers[4] = 2`

main.java

```
1  int solution
2
3  }
4
```

TESTS CUSTOM

MORE

app.co

Search

DESC

HISTORY

RULES

README

SETTINGS

Codewriting

Given an array of integers `numbers`, your task is to count all numbers that are **not** equal to `numbers[0]` or `numbers[1]`, if they exist in the array.

Assume that array indices are 0-based.

Note: You are not expected to provide the most optimal solution, but a solution with time complexity not worse than $O(numbers.length^2)$ *will fit within the execution time limit.*

Example

- For `numbers = [4, 3, 2, 3, 2, 5, 4, 3]`, the output should be `solution(numbers) = 3`.

  Explanation:

  `numbers[0]` and `numbers[1]` are `4` and `3` respectively. There are three elements in `numbers` that are not equal to `4` or `3`:

  - `numbers[2] = 2`
  - `numbers[4] = 2`

main.java

```
1  int solution
2
3  }
4
```

TESTS CUSTOM

MORE

app.co

Search

Codewriting

An array is considered to follow an *alternating parity pattern* if it does **not** contain two consecutive integers of the same parity. In other words, all consecutive integers in the array will never be two even numbers nor two odd numbers.

Given an array of non-negative integers `numbers`, determine whether there are elements which break the *alternating parity pattern*. Return the index of the first element where the alternating parity pattern breaks, or `-1` if there are no such elements.

Note: You are not expected to provide the most optimal solution, but a solution with time complexity not worse than `O(numbers.length²)` *will fit within the execution time limit.*

Example

- For `numbers = [1, 2, 5, 3, 6]`, the output should be `solution(numbers) = 3`.

  Since both `numbers[2] = 5` and `numbers[3] = 3` are odd, the first index where the alternating parity pattern breaks is `numbers[3]`

ASSESSMENT LINK | 2 × Signed Out × CodeSignal

app.codesignal.com

TO QUESTIONS

Codewriting

An array is considered to follow an *alternating parity pattern* if it does **not** contain two consecutive integers of the same parity. In other words, all consecutive integers in the array will never be two even numbers nor two odd numbers.

Given an array of non-negative integers `numbers`, determine whether there are elements which break the *alternating parity pattern*. Return the index of the first element where the alternating parity pattern breaks, or `-1` if there are no such elements.

Note: You are not expected to provide the most optimal solution, but a solution with time complexity not worse than $O(numbers.length^2)$ will fit within the execution time limit.

Example

- For `numbers = [1, 2, 5, 3, 6]`, the output should be `solution(numbers) = 3`.

  Since both `numbers[2] = 5` and `numbers[3] = 3` are odd, the first index where the alternating parity pattern breaks is `numbers[3]`.

Given an integer `n`, your task is to create a square frame of size `n`, represented as an array of strings. The frame should consist of empty space, enclosed by lines made of `*` characters as follows:

```
********
*      *
*      *
*      *
*      *
*      *
*      *
********
```

Note: You are not expected to provide the most optimal solution, but a solution with time complexity not worse than $O(n^3)$ will fit within the execution time limit.

Example

- For `n = 8`, the output should be

```
solution(n) = [
  "********",
  "*      *",
  "*      *",
  "*      *",
  "*      *",
  "*      *",
  "*      *",
```

main.java

app.codesignal.co

Search

ly cloudy

Given an integer `n`, your task is to create a square frame of size `n`, represented as an array of strings. The frame should consist of empty space, enclosed by lines made of `*` characters as follows:

```
********
*      *
*      *
*      *
*      *
*      *
*      *
********
```

Note: You are not expected to provide the most optimal solution, but a solution with time complexity not worse than $O(n^3)$ *will fit within the execution time limit.*

Example

- For `n = 8`, the output should be

```
solution(n) = [
  "********",
  "*      *",
  "*      *",
  "*      *",
  "*      *",
  "*      *",
  "*      *",
  "********"
]
```

main.java

app.codesignal.co

Search

ly cloudy

Your task is to find the time when the last person will be processed. The output should be time in seconds since the starting time of the event.

Notes:

- The queue size is calculated by the number of people waiting to start their ID check, not including the person who is already in the process of an ID check.
- If a new person arrives at the same moment as when another person completes their ID check, the first person waiting in the queue should have their ID checked first, and the new person should wait in the queue.

Note: You are not expected to provide the most optimal solution, but a solution with time complexity not worse than $O(times.length^2)$ *will fit within the execution time limit.*

Example

Codewriting

Imagine an exclusive event that many people wish to attend. The event starts at time `0`. For every person attending, you are given a time in seconds (since the start time of the event) representing when they arrived at the event. However, entry into this event requires an identification check which takes time, so people may need to wait in a queue to enter. Specifically, it takes 5 minutes to do an ID check for every attendee. Additionally, if the person arrives at the event and sees that there are more than *10* people in the queue, they leave immediately.

Your task is to find the time when the last person will be processed. The output should be time in seconds since the starting time of the event.

Notes:

- The queue size is calculated by the number of people waiting to start their ID check, not including

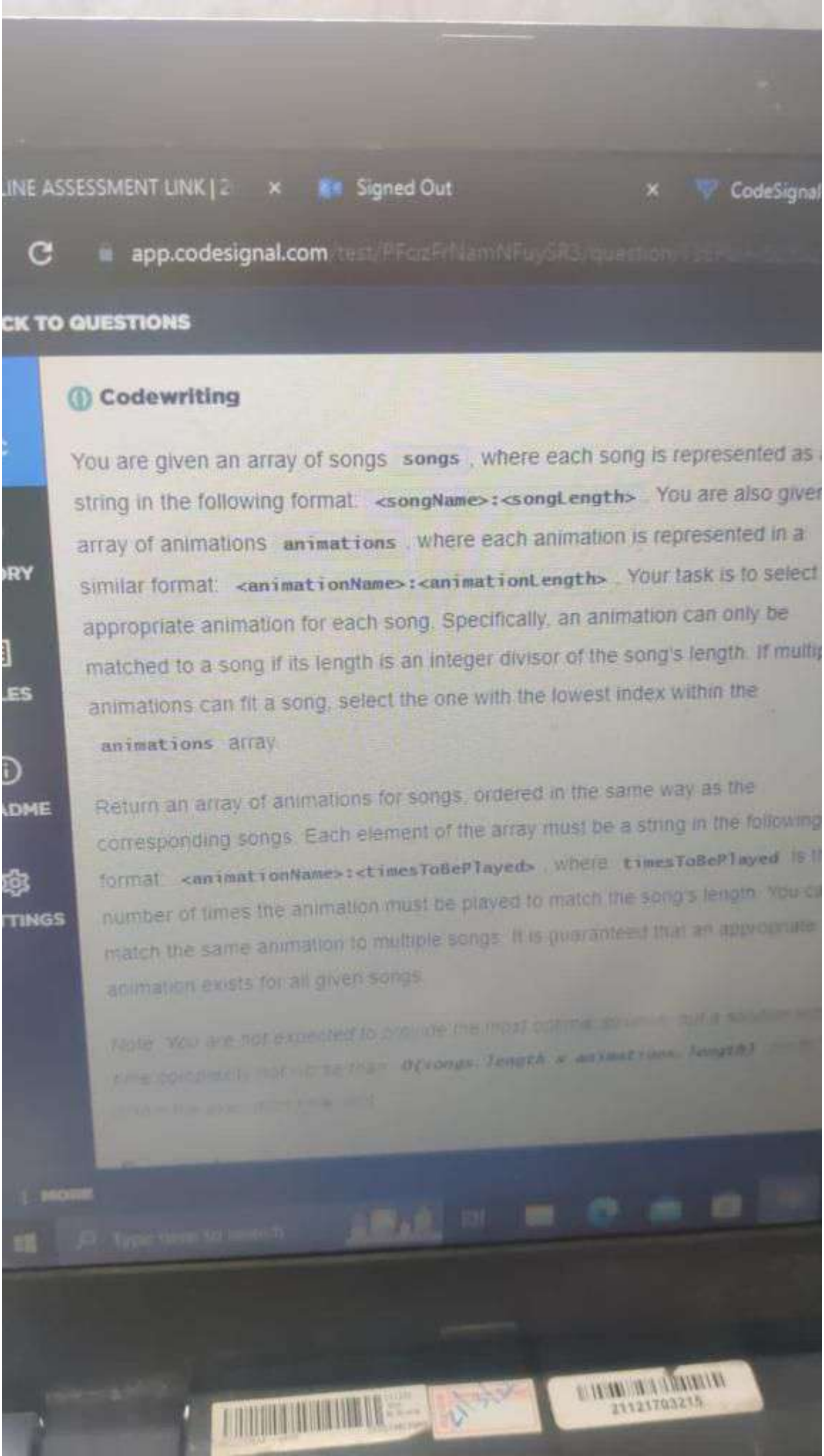
Note: You are not expected to provide the most optimal solution, but a solution with time complexity not worse than `O(times.length^2)` *will fit within the execution time limit.*

Example

- For `times = [1, 6, 9, 502]`, the output should be `solution(times) = 1201`.

  Let's consider this scenario:

  - The first person arrives at time `1`, and there is no one in the queue, so they start the ID check. Queue - `[]`.
  - The second person arrives at time `6`, and there is no one in the queue, but there is one person getting their ID checked, so they wait in the queue. Queue - `[2]`.
  - The third person arrives at time `9`, and there is one person in the queue, so they also wait


LINE ASSESSMENT LINK | 2
Signed Out
CodeSignal
app.codesignal.com
CK TO QUESTIONS
Codewriting
You are given an array of songs songs , where each song is represented as
string in the following format: <songName>:<songLength> . You are also given
array of animations animations , where each animation is represented in a
similar format: <animationName>:<animationLength> . Your task is to select
appropriate animation for each song. Specifically, an animation can only be
matched to a song if its length is an integer divisor of the song's length. If multi
animations can fit a song, select the one with the lowest index within the
animations array.
Return an array of animations for songs, ordered in the same way as the
corresponding songs. Each element of the array must be a string in the following
format: <animationName>:<timesToBePlayed> , where timesToBePlayed is t
number of times the animation must be played to match the song's length. You c
match the same animation to multiple songs. It is guaranteed that an appropriate
animation exists for all given songs.

Codewriting

You are given an array of songs `songs`, where each song is represented as a string in the following format: `<songName>:<songLength>`. You are also given an array of animations `animations`, where each animation is represented in a similar format: `<animationName>:<animationLength>`. Your task is to select an appropriate animation for each song. Specifically, an animation can only be matched to a song if its length is an integer divisor of the song's length. If multiple animations can fit a song, select the one with the lowest index within the `animations` array.

Return an array of animations for songs, ordered in the same way as the corresponding songs. Each element of the array must be a string in the following format: `<animationName>:<timesToBePlayed>`, where `timesToBePlayed` is the number of times the animation must be played to match the song's length. You can match the same animation to multiple songs. It is guaranteed that an appropriate animation exists for all given songs.

`1 ≤ songLength ≤ 50000`.

- **[input] array.string animations**

  An array of strings representing animations, with each animation in this format: `"<animationName>: <animationLength>"`. `<animationName>` consists only of English letters, and `<animationLength>` consists of integers values.

  Guaranteed constraints:

  `1 ≤ animations.length ≤ 100`,

  `1 ≤ animations[i].length ≤ 16`,

  `1 ≤ animationLength ≤ 50000`

- **[output] array.string**

  An array of strings representing the selected animation for each song in `songs`, and the number of times they need to be played to match the corresponding song length. Each element should be in this format: `"<animationName>:<timesToBePlayed>"`

Note: You are not expected to provide the most optimal solution, but a solution with time complexity not worse than `O(songs.length x animations.length)` *will fit within the execution time limit.*

Example

- For `songs = ["notion:180", "voyage:185", "sample:180"]` and `animations = ["circles:360", "squares:180", "lines:37"]`, the output should be `solution(songs, animations) = ["squares:1", "lines:5", "squares:1"]`.

  Explanation:

  - For `songs[0] = "notion:180"`, `animation[0] = "circles:360"` won't fit because the animation's length shouldn't exceed the song's length.
  - For `songs[0] = "notion:180"`, `animation[1] = "squares:180"` is appropriate because their lengths are equal. So, the animation for the first

E ASSESSMENT LINK | : × Signed Out × CodeSignal

app.codesignal.com

K TO QUESTIONS

Example

- For `songs = ["notion:180", "voyage:185", "sample:180"]` and `animations = ["circles:360", "squares:180", "lines:37"]`, the output should be `solution(songs, animations) = ["squares:1", "lines:5", "squares:1"]`.

  Explanation:

  - For `songs[0] = "notion:180"`, `animation[0] = "circles:360"` won't fit because the animation's length shouldn't exceed the song's length.
  - For `songs[0] = "notion:180"`, `animation[1] = "squares:180"` is appropriate because their lengths are equal. So, the animation for the first song is `animation[1] = "squares:180"`, which will be played just once.
  - For `songs[1] = "voyage:185"`, `animation[0] = "circles:360` won't fit because the animation's length shouldn't exceed the song's length.
  - For `songs[1] = "voynge:185"`, `animation[1] = "squares:180` won't fit because its length of `180` is not an integer divisor of the song's length of `185`.

RY
ES
DME
TINGS
MORE

Type here to search

- **[input] array.string songs**

  An array of strings representing songs, with each song in this format: `"<songName>:<songLength>"`. `<songName>` consists of English letters, and `<songLength>` consists of integer values.

  Guaranteed constraints:

  `1 ≤ songs.length ≤ 100`,

  `1 ≤ songs[i].length ≤ 16`,

  `1 ≤ songLength ≤ 50000`

- **[input] array.string animations**

  An array of strings representing animations, with each animation in this format: `"<animationName>:<animationLength>"`. `<animationName>` consists only of English letters, and `<animationLength>` consists of integers values.

  Guaranteed constraints:

  `1 ≤ animations.length ≤ 100`,

  `1 ≤ animations[i].length ≤ 16`,

  `1 ≤ animationLength ≤ 50000`

E ASSESSMENT LINK | Signed Out CodeSignal

app.codesignal.com

K TO QUESTIONS

Example

- For `songs = ["notion:180", "voyage:185", "sample:180"]` and `animations = ["circles:360", "squares:180", "lines:37"]`, the output should be `solution(songs, animations) = ["squares:1", "lines:5", "squares:1"]`.

  Explanation:

  - For `songs[0] = "notion:180"`, `animation[0] = "circles:360"` won't fit because the animation's length shouldn't exceed the song's length.
  - For `songs[0] = "notion:180"`, `animation[1] = "squares:180"` is appropriate because their lengths are equal. So, the animation for the first song is `animation[1] = "squares:180"`, which will be played just once.
  - For `songs[1] = "voyage:185"`, `animation[0] = "circles:360` won't fit because the animation's length shouldn't exceed the song's length.
  - For `songs[1] = "voyage:185"` `animation[1] = "squares:180` won't fit because its length of `180` is not an integer divisor of the song's length of `185`.

MORE

Type here to search

ESSMENT LINK | 2 × Signed Out × CodeSignal

app.codesignal.com/test/PFozFrNamNFuy3R3/question/

QUESTIONS

- **[input] array.string songs**

  An array of strings representing songs, with each song in this format: `"<songName>:<songLength>"`. `<songName>` consists of English letters, and `<songLength>` consists of integer values.

  Guaranteed constraints:

  `1 ≤ songs.length ≤ 100`,

  `1 ≤ songs[i].length ≤ 16`,

  `1 ≤ songLength ≤ 50000`

- **[input] array.string animations**

  An array of strings representing animations, with each animation in this format: `"<animationName>:<animationLength>"`. `<animationName>` consists only of English letters, and `<animationLength>` consists of integers values.

  Guaranteed constraints:

  `1 ≤ animations.length ≤ 100`,

  `1 ≤ animations[i].length ≤ 16`,

  `1 ≤ animationLength ≤ 50000`

Type here to search

Guaranteed constraints:

`1 ≤ animations.length ≤ 100`.

`1 ≤ animations[i].length ≤ 16`.

`1 ≤ animationLength ≤ 50000`.

- **[output] array.string**

  An array of strings representing the selected animation for each song in `songs`, and the number of times they need to be played to match the corresponding song length. Each element should be in this format "`<animationName>:<timesToBePlayed>`".

[C] Syntax Tips

```
// Prints help message to the console
// Returns a string
char * helloWorld(char * name) {
    char * answer = malloc(strlen(name) + 8);
    printf("This prints to the console when you Run Tests");
    strcpy(answer, "Hello, ");
    strcat(answer, name);
    return answer;
}
```

ESSMENT LINK | 2 × Signed Out × CodeSignal

app.codesignal.com

QUESTIONS

Guaranteed constraints:

`1 ≤ animations.length ≤ 100`,

`1 ≤ animations[i].length ≤ 16`,

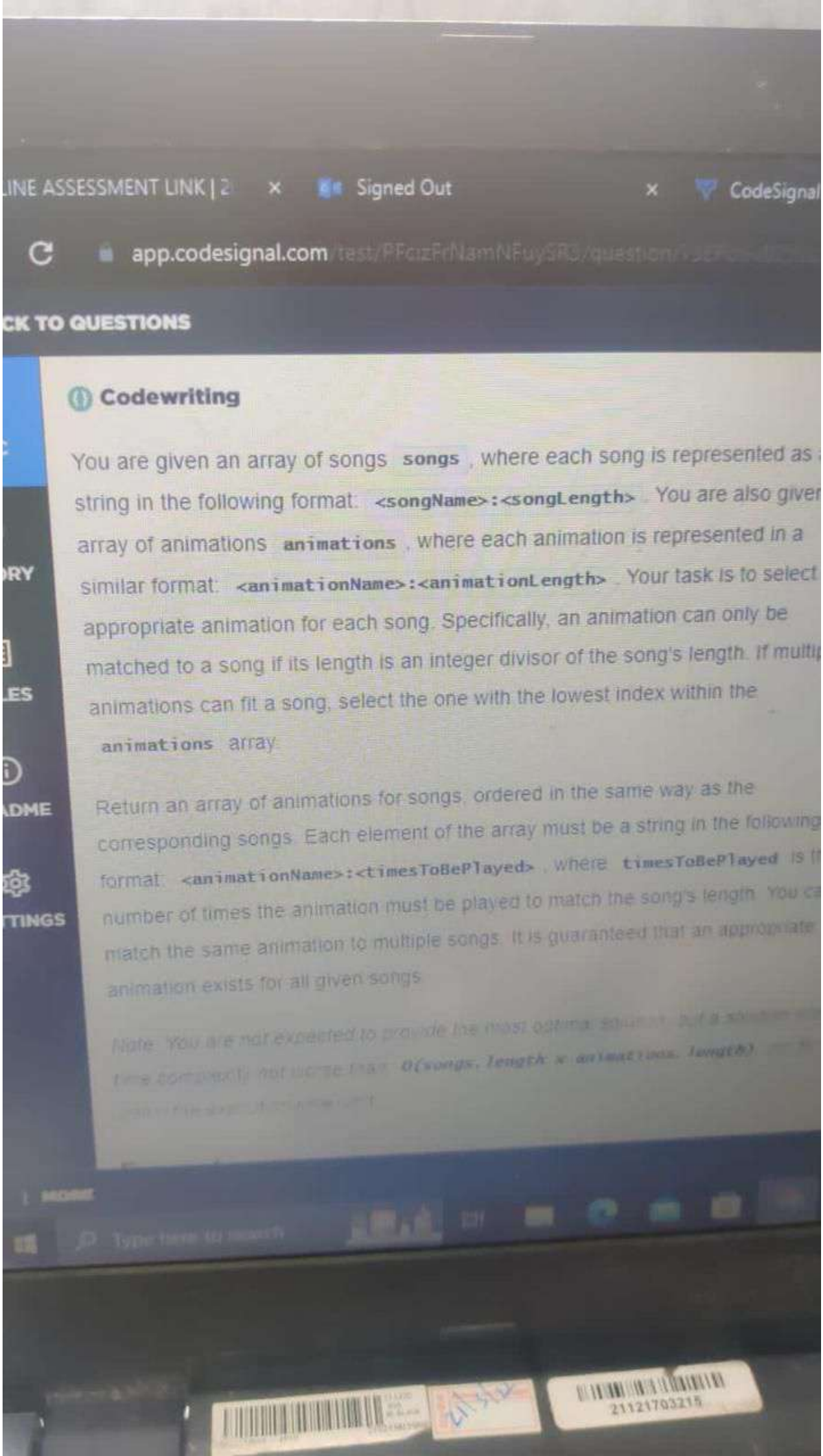
`1 ≤ animationLength ≤ 50000`.

- **[output] array.string**

  An array of strings representing the selected animation for each song in `songs`, and the number of times they need to be played to match the corresponding song length. Each element should be in this format "`<animationName>:<timesToBePlayed>`"

[C] Syntax Tips

```
// Prints help message to the console
// Returns a string
char * helloWorld(char * name) {
    char * answer = malloc(strlen(name) + 8);
    printf("This prints to the console when you Run Tests");
    strcpy(answer, "Hello, ");
    strcat(answer, name);
    return answer;
}
```


LINE ASSESSMENT LINK | 2
Signed Out
CodeSignal
app.codesignal.com
CK TO QUESTIONS
Codewriting
You are given an array of songs songs , where each song is represented as
string in the following format: <songName>:<songLength> . You are also given
array of animations animations , where each animation is represented in a
similar format: <animationName>:<animationLength> . Your task is to select
appropriate animation for each song. Specifically, an animation can only be
matched to a song if its length is an integer divisor of the song's length. If multi
animations can fit a song, select the one with the lowest index within the
animations array.
Return an array of animations for songs, ordered in the same way as the
corresponding songs. Each element of the array must be a string in the following
format: <animationName>:<timesToBePlayed> , where timesToBePlayed is t
number of times the animation must be played to match the song's length. You ca
match the same animation to multiple songs. It is guaranteed that an appropriate
animation exists for all given songs.
Note: You are not expected to provide the most optimal solution, but a solution
time complexity not worse than O(songs.length * animations.length)
21121703215

Note: You are not expected to provide the most optimal solution, but a solution with time complexity not worse than `O(songs.length x animations.length)` *will fit within the execution time limit.*

Example

- For `songs = ["notion:180", "voyage:185", "sample:180"]` and `animations = ["circles:360", "squares:180", "lines:37"]`, the output should be `solution(songs, animations) = ["squares:1", "lines:5", "squares:1"]`.

  Explanation:

  - For `songs[0] = "notion:180"`, `animation[0] = "circles:360"` won't fit because the animation's length shouldn't exceed the song's length.
  - For `songs[0] = "notion:180"`, `animation[1] = "squares:180"` is appropriate because their lengths are equal. So, the animation for the first

`1 ≤ songLength ≤ 50000`.

- **[input] array.string animations**

  An array of strings representing animations, with each animation in this format: `"<animationName>:<animationLength>"`. `<animationName>` consists only of English letters, and `<animationLength>` consists of integers values.

  Guaranteed constraints:

  `1 ≤ animations.length ≤ 100`,

  `1 ≤ animations[i].length ≤ 16`,

  `1 ≤ animationLength ≤ 50000`.

- **[output] array.string**

  An array of strings representing the selected animation for each song in `songs`, and the number of times they need to be played to match the corresponding song length. Each element should be in this format: `"<animationName>:<timesToBePlayed>"`

Codewriting

You are given an array of songs `songs`, where each song is represented as a string in the following format: `<songName>:<songLength>`. You are also given an array of animations `animations`, where each animation is represented in a similar format: `<animationName>:<animationLength>`. Your task is to select an appropriate animation for each song. Specifically, an animation can only be matched to a song if its length is an integer divisor of the song's length. If multiple animations can fit a song, select the one with the lowest index within the `animations` array.

Return an array of animations for songs, ordered in the same way as the corresponding songs. Each element of the array must be a string in the following format: `<animationName>:<timesToBePlayed>`, where `timesToBePlayed` is the number of times the animation must be played to match the song's length. You can match the same animation to multiple songs. It is guaranteed that an appropriate animation exists for all given songs.

TO QUESTIONS

Also note that you are not expected to provide the most optimal solution, but a solution with time complexity not worse than $O(logs.length^2)$ *will fit within the execution time limit.*

Example

- For

```
logs = [
  "switch branch1",
  "push file1",
  "push file2",
  "push file1",
  "switch branch2",
  "switch issue2",
  "push file1",
  "push file2",
  "push file3"
]
```

the output should be `solution(logs) = "issue2"`.

Explanation:

- Command 0: `switch branch1` changes the current branch to `branch1`
- Commands 1, 2: The next two logs push `file1` and `file2`

CodeSignal

BACK TO QUESTIONS

Codewriting

You are developing a simple version control system. Your system can support branches, and you can switch between branches or update the files in a branch using the following actions:

- `switch <branch name>` – switches to the branch with given name.
- `push <file name>` – pushes a file to the current branch. **Note, that if the file was pushed to the current branch before, then the number of files should not be changed.**

Your task is to process all the logs and return the branch name that contains the largest number of files at the end. It is guaranteed that there is no tie.

Notes:

- It is guaranteed that the first command is `switch`.
- It is guaranteed that there is at least one `push` command.

Also note that you are not expected to provide the most optimal solution, but a solution with time complexity not worse than $O(logs.length^2)$ *will fit within the execution time limit.*

Example

eaking news
t caught up

Search

CodeSignal

BACK TO QUESTIONS

Codewriting

You are developing a simple version control system. Your system can support branches, and you can switch between branches or update the files in a branch using the following actions:

- `switch <branch name>` – switches to the branch with given name.
- `push <file name>` – pushes a file to the current branch. **Note, that if the file was pushed to the current branch before, then the number of files should not be changed.**

Your task is to process all the logs and return the branch name that contains the largest number of files at the end. It is guaranteed that there is no tie.

Notes:

- It is guaranteed that the first command is `switch`.
- It is guaranteed that there is at least one `push` command.

Also note that you are not expected to provide the most optimal solution, but a solution with time complexity not worse than $O(logs.length^2)$ *will fit within the execution time limit.*

Example

eaking news
t caught up

Search

Also note that you are not expected to provide the most optimal solution, but a solution with time complexity not worse than $O(logs.length^2)$ *will fit within the execution time limit.*

Example

- For

```
logs = [
  "switch branch1",
  "push file1",
  "push file2",
  "push file1",
  "switch branch2",
  "switch issue2",
  "push file1",
  "push file2",
  "push file3"
]
```

the output should be `solution(logs) = "issue2"`.

Explanation:

- Command 0: `switch branch1` changes the current branch to `branch1`
- Commands 1, 2: The next two logs push `file1` and `file2`

ONLINE ASSESSMENT LINK | 2 × Signed Out × Cod

app.codesignal.com/test/PFozFrNamNPuySR3/

BACK TO QUESTIONS

DESC

STORY

RULES

README

SETTINGS

Input/Output

- [execution time limit] 0.5 seconds (c)
- [memory limit] 1 GB
- [input] array.integer numbers

  An array of integers

  Guaranteed constraints:

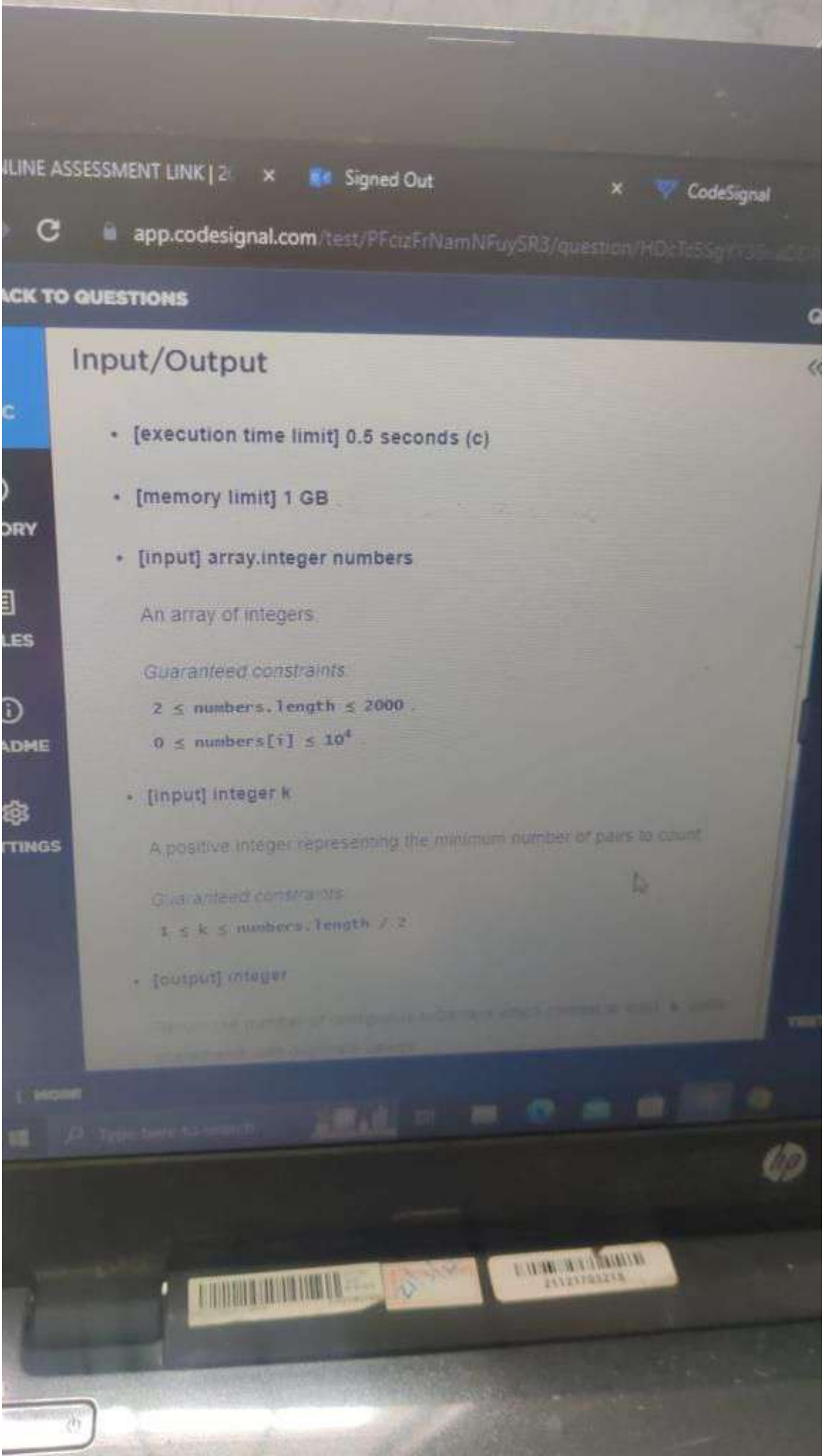
  $1 \leq$ numbers.length ≤ 1000

  $0 \leq$ numbers[i] $\leq 10^9$

- [output] integer

  Return the index of the first element where the absolute parity sorted breaks, or -1 if there are no such elements.

[C] Syntax Tips


NLINE ASSESSMENT LINK | 2
Signed Out
CodeSignal
app.codesignal.com/test/PFctzFrNamNFuySR3/question/HDcTcSgY
ACK TO QUESTIONS
Input/Output
[execution time limit] 0.5 seconds (c)
[memory limit] 1 GB
[input] array.integer numbers
An array of integers.
Guaranteed constraints:
2 ≤ numbers.length ≤ 2000,
0 ≤ numbers[i] ≤ 10^4.
[input] integer k
A positive integer representing the minimum number of pairs to count.
Guaranteed constraints:
1 ≤ k ≤ numbers.length / 2.
[output] integer

ONLINE ASSESSMENT LINK | 2 × Signed Out × Cod

app.codesignal.com/test/PFcizFrNamNFuySR3/question

BACK TO QUESTIONS

DESC

STORY

RULES

README

SETTINGS

Input/Output

- [execution time limit] 0.5 seconds (c)
- [memory limit] 1 GB
- [input] array.integer numbers

  An array of integers.

  Guaranteed constraints:

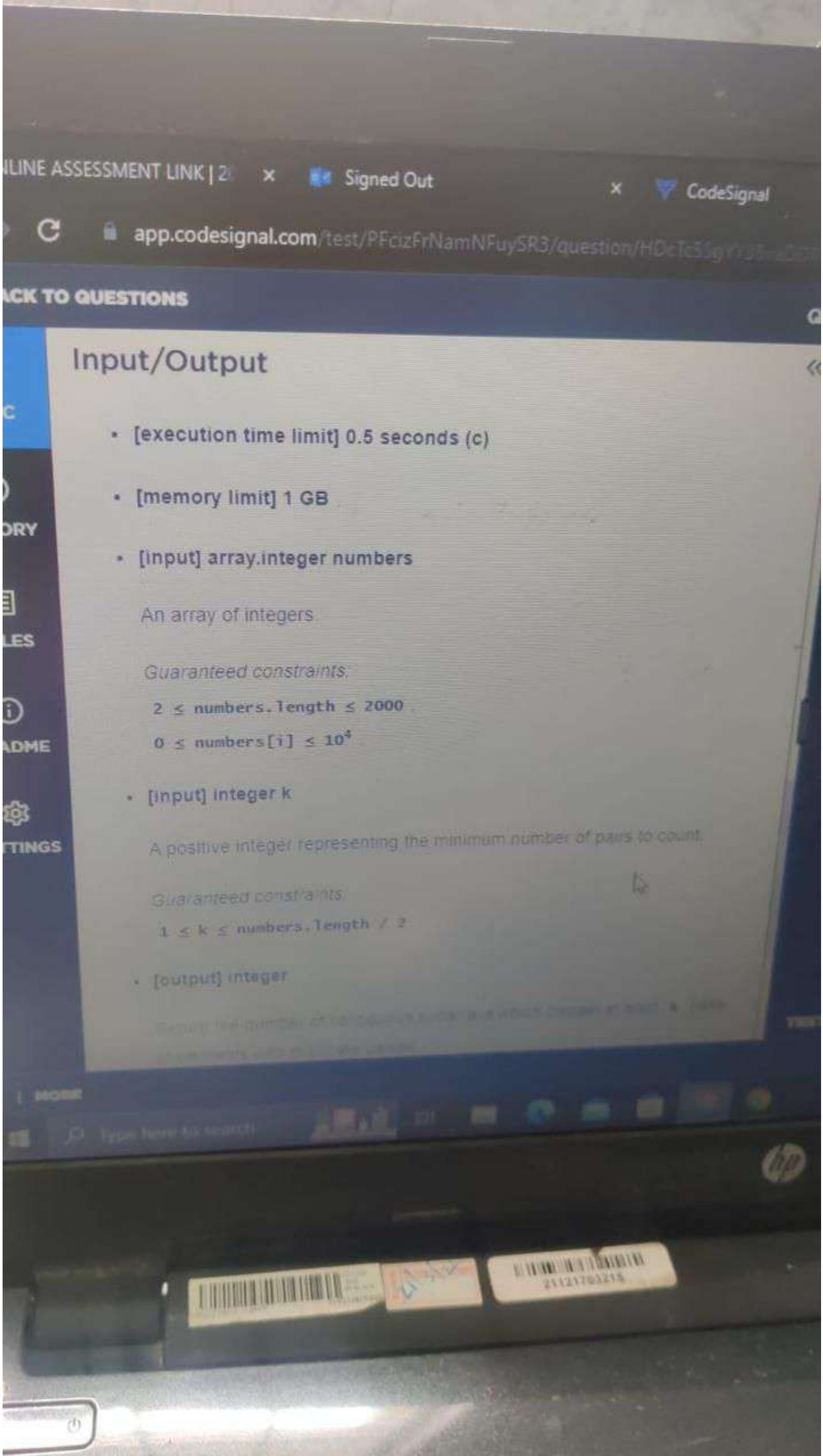
  $1 \leq$ numbers.length ≤ 1000,

  $0 \leq$ numbers[i] $\leq 10^9$.

- [output] integer

  Return the index of the first element where the alternating parity pattern breaks, or -1 if there are no such elements.

[C] Syntax Tips


NLINE ASSESSMENT LINK | 2
Signed Out
CodeSignal
app.codesignal.com/test/PFcizFrNamNFuySR3/question/HDcTcSSgY
ACK TO QUESTIONS
Input/Output
[execution time limit] 0.5 seconds (c)
[memory limit] 1 GB
[input] array.integer numbers
An array of integers.
Guaranteed constraints:
2 ≤ numbers.length ≤ 2000,
0 ≤ numbers[i] ≤ 10^4.
[input] integer k
A positive integer representing the minimum number of pairs to count.
Guaranteed constraints:
1 ≤ k ≤ numbers.length / 2.
[output] integer