Problem 1894: Find the Student that Will Replace the Chalk

Problem Information

Difficulty: Medium
Acceptance Rate: 0.00%
Paid Only: No

Problem Description

There are

n

students in a class numbered from

0

to

n - 1

. The teacher will give each student a problem starting with the student number

0

, then the student number

1

, and so on until the teacher reaches the student number

n - 1

. After that, the teacher will restart the process, starting with the student number

0

again.

You are given a

0-indexed

integer array

chalk

and an integer

k

. There are initially

k

pieces of chalk. When the student number

i

is given a problem to solve, they will use

chalk[i]

pieces of chalk to solve that problem. However, if the current number of chalk pieces is

strictly less

than

chalk[i]

, then the student number

i

will be asked to

replace

the chalk.

Return

the

index

of the student that will

replace

the chalk pieces

.

Example 1:

Input:

chalk = [5,1,5], k = 22

Output:

0

Explanation:

The students go in turns as follows: - Student number 0 uses 5 chalk, so k = 17. - Student number 1 uses 1 chalk, so k = 16. - Student number 2 uses 5 chalk, so k = 11. - Student number 0 uses 5 chalk, so k = 6. - Student number 1 uses 1 chalk, so k = 5. - Student number 2 uses 5 chalk, so k = 0. Student number 0 does not have enough chalk, so they will have to replace it.

Example 2:

Input:

chalk = [3,4,1,2], k = 25

Output:

1

Explanation:

The students go in turns as follows: - Student number 0 uses 3 chalk so k = 22. - Student number 1 uses 4 chalk so k = 18. - Student number 2 uses 1 chalk so k = 17. - Student number 3 uses 2 chalk so k = 15. - Student number 0 uses 3 chalk so k = 12. - Student number 1 uses 4 chalk so k = 8. - Student number 2 uses 1 chalk so k = 7. - Student number 3 uses 2 chalk so k = 5. - Student number 0 uses 3 chalk so k = 2. Student number 1 does not have enough chalk, so they will have to replace it.

Constraints:

chalk.length == n

1 <= n <= 10

5

1 <= chalk[i] <= 10

5

1 <= k <= 10

9

Code Snippets

C++:

```
class Solution {
public:
int chalkReplacer(vector<int>& chalk, int k) {

}
};
```

Java:

```
class Solution {
public int chalkReplacer(int[] chalk, int k) {

}
}
```

Python3:

```
class Solution:
def chalkReplacer(self, chalk: List[int], k: int) -> int:
```

Python:

```
class Solution(object):
def chalkReplacer(self, chalk, k):
"""
:type chalk: List[int]
:type k: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number[]} chalk
* @param {number} k
* @return {number}
*/
var chalkReplacer = function(chalk, k) {

};
```

TypeScript:

```
function chalkReplacer(chalk: number[], k: number): number {

};
```

C#:

```
public class Solution {
public int ChalkReplacer(int[] chalk, int k) {

}
}
```

C:

```
int chalkReplacer(int* chalk, int chalkSize, int k) {

}
```

Go:

```
func chalkReplacer(chalk []int, k int) int {

}
```

Kotlin:

```
class Solution {
fun chalkReplacer(chalk: IntArray, k: Int): Int {

}
}
```

Swift:

```
class Solution {
func chalkReplacer(_ chalk: [Int], _ k: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn chalk_replacer(chalk: Vec<i32>, k: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[]} chalk
# @param {Integer} k
# @return {Integer}
def chalk_replacer(chalk, k)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $chalk
* @param Integer $k
* @return Integer
*/
function chalkReplacer($chalk, $k) {

}
}
```

Dart:

```
class Solution {
int chalkReplacer(List<int> chalk, int k) {

}
}
```

Scala:

```
object Solution {
def chalkReplacer(chalk: Array[Int], k: Int): Int = {

}
```

```
}
```

Elixir:

```
defmodule Solution do
@spec chalk_replacer(chalk :: [integer], k :: integer) :: integer
def chalk_replacer(chalk, k) do

end
end
```

Erlang:

```
-spec chalk_replacer(Chalk :: [integer()], K :: integer()) -> integer().
chalk_replacer(Chalk, K) ->
.
```

Racket:

```
(define/contract (chalk-replacer chalk k)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find the Student that Will Replace the Chalk
* Difficulty: Medium
* Tags: array, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
int chalkReplacer(vector<int>& chalk, int k) {
```

```
}
};
```

Java Solution:

```
/**
* Problem: Find the Student that Will Replace the Chalk
* Difficulty: Medium
* Tags: array, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public int chalkReplacer(int[] chalk, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Find the Student that Will Replace the Chalk
Difficulty: Medium
Tags: array, search

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def chalkReplacer(self, chalk: List[int], k: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def chalkReplacer(self, chalk, k):
"""
:type chalk: List[int]
:type k: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Find the Student that Will Replace the Chalk
* Difficulty: Medium
* Tags: array, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {number[]} chalk
* @param {number} k
* @return {number}
*/
var chalkReplacer = function(chalk, k) {

};
```

TypeScript Solution:

```
/**
* Problem: Find the Student that Will Replace the Chalk
* Difficulty: Medium
* Tags: array, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function chalkReplacer(chalk: number[], k: number): number {
```

```
};
```

C# Solution:

```
/*
* Problem: Find the Student that Will Replace the Chalk
* Difficulty: Medium
* Tags: array, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public int ChalkReplacer(int[] chalk, int k) {

}
}
```

C Solution:

```
/*
* Problem: Find the Student that Will Replace the Chalk
* Difficulty: Medium
* Tags: array, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

int chalkReplacer(int* chalk, int chalkSize, int k) {

}
```

Go Solution:

```
// Problem: Find the Student that Will Replace the Chalk
// Difficulty: Medium
```

```
// Tags: array, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func chalkReplacer(chalk []int, k int) int {

}
```

Kotlin Solution:

```
class Solution {
fun chalkReplacer(chalk: IntArray, k: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func chalkReplacer(_ chalk: [Int], _ k: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Find the Student that Will Replace the Chalk
// Difficulty: Medium
// Tags: array, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn chalk_replacer(chalk: Vec<i32>, k: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} chalk
# @param {Integer} k
# @return {Integer}
def chalk_replacer(chalk, k)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $chalk
* @param Integer $k
* @return Integer
*/
function chalkReplacer($chalk, $k) {

}
}
```

Dart Solution:

```
class Solution {
int chalkReplacer(List<int> chalk, int k) {

}
}
```

Scala Solution:

```
object Solution {
def chalkReplacer(chalk: Array[Int], k: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec chalk_replacer(chalk :: [integer], k :: integer) :: integer
def chalk_replacer(chalk, k) do

end
end
```

Erlang Solution:

```
-spec chalk_replacer(Chalk :: [integer()], K :: integer()) -> integer().
chalk_replacer(Chalk, K) ->
.
```

Racket Solution:

```
(define/contract (chalk-replacer chalk k)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```