Problem 2678: Number of Senior Citizens

Problem Information

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

Problem Description

You are given a

0-indexed

array of strings

details

. Each element of

details

provides information about a given passenger compressed into a string of length

15

. The system is such that:

The first ten characters consist of the phone number of passengers.

The next character denotes the gender of the person.

The following two characters are used to indicate the age of the person.

The last two characters determine the seat allotted to that person.

Return

the number of passengers who are

strictly

more than 60 years old

.

Example 1:

Input:

details = ["7868190130M7522","5303914400F9211","9273338290F4010"]

Output:

2

Explanation:

The passengers at indices 0, 1, and 2 have ages 75, 92, and 40. Thus, there are 2 people who are over 60 years old.

Example 2:

Input:

details = ["1313579440F2036","2921522980M5644"]

Output:

0

Explanation:

None of the passengers are older than 60.

Constraints:

1 <= details.length <= 100

details[i].length == 15

details[i] consists of digits from '0' to '9'.

details[i][10] is either 'M' or 'F' or 'O'.

The phone numbers and seat numbers of the passengers are distinct.

Code Snippets

C++:

```
class Solution {
public:
int countSeniors(vector<string>& details) {

}
};
```

Java:

```
class Solution {
public int countSeniors(String[] details) {

}
}
```

Python3:

```
class Solution:
def countSeniors(self, details: List[str]) -> int:
```

Python:

```
class Solution(object):
def countSeniors(self, details):
```

```
"""
:type details: List[str]
:rtype: int
"""
```

JavaScript:

```
/**
* @param {string[]} details
* @return {number}
*/
var countSeniors = function(details) {

};
```

TypeScript:

```
function countSeniors(details: string[]): number {

};
```

C#:

```
public class Solution {
public int CountSeniors(string[] details) {

}
}
```

C:

```
int countSeniors(char** details, int detailsSize) {

}
```

Go:

```
func countSeniors(details []string) int {

}
```

Kotlin:

```
class Solution {
fun countSeniors(details: Array<String>): Int {

}
}
```

Swift:

```
class Solution {
func countSeniors(_ details: [String]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn count_seniors(details: Vec<String>) -> i32 {

}
}
```

Ruby:

```
# @param {String[]} details
# @return {Integer}
def count_seniors(details)

end
```

PHP:

```
class Solution {

/**
* @param String[] $details
* @return Integer
*/
function countSeniors($details) {

}
}
```

Dart:

```
class Solution {
int countSeniors(List<String> details) {

}
}
```

Scala:

```
object Solution {
def countSeniors(details: Array[String]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec count_seniors(details :: [String.t]) :: integer
def count_seniors(details) do

end
end
```

Erlang:

```
-spec count_seniors(Details :: [unicode:unicode_binary()]) -> integer().
count_seniors(Details) ->
.
```

Racket:

```
(define/contract (count-seniors details)
(-> (listof string?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Number of Senior Citizens
* Difficulty: Easy
* Tags: array, string
*
* 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 countSeniors(vector<string>& details) {

}
};
```

Java Solution:

```
/**
* Problem: Number of Senior Citizens
* Difficulty: Easy
* Tags: array, string
*
* 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 countSeniors(String[] details) {

}
}
```

Python3 Solution:

```
"""
Problem: Number of Senior Citizens
Difficulty: Easy
Tags: array, string
```

```
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 countSeniors(self, details: List[str]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def countSeniors(self, details):
"""
:type details: List[str]
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Number of Senior Citizens
* Difficulty: Easy
* Tags: array, string
*
* 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 {string[]} details
* @return {number}
*/
var countSeniors = function(details) {

};
```

TypeScript Solution:

```
/**
* Problem: Number of Senior Citizens
* Difficulty: Easy
* Tags: array, string
*
* 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 countSeniors(details: string[]): number {

};
```

C# Solution:

```
/*
* Problem: Number of Senior Citizens
* Difficulty: Easy
* Tags: array, string
*
* 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 CountSeniors(string[] details) {

}
}
```

C Solution:

```
/*
* Problem: Number of Senior Citizens
* Difficulty: Easy
* Tags: array, string
*
* 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 countSeniors(char** details, int detailsSize) {

}
```

Go Solution:

```
// Problem: Number of Senior Citizens
// Difficulty: Easy
// Tags: array, string
//
// 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 countSeniors(details []string) int {

}
```

Kotlin Solution:

```
class Solution {
fun countSeniors(details: Array<String>): Int {

}
}
```

Swift Solution:

```
class Solution {
func countSeniors(_ details: [String]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Number of Senior Citizens
// Difficulty: Easy
// Tags: array, string
```

```
//
// 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 count_seniors(details: Vec<String>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String[]} details
# @return {Integer}
def count_seniors(details)

end
```

PHP Solution:

```
class Solution {

/**
* @param String[] $details
* @return Integer
*/
function countSeniors($details) {

}
}
```

Dart Solution:

```
class Solution {
int countSeniors(List<String> details) {

}
}
```

Scala Solution:

```
object Solution {
def countSeniors(details: Array[String]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec count_seniors(details :: [String.t]) :: integer
def count_seniors(details) do

end
end
```

Erlang Solution:

```
-spec count_seniors(Details :: [unicode:unicode_binary()]) -> integer().
count_seniors(Details) ->
.
```

Racket Solution:

```
(define/contract (count-seniors details)
(-> (listof string?) exact-integer?)
)
```