Problem 379: Design Phone Directory

Problem Information

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

Problem Description

Design a phone directory that initially has

maxNumbers

empty slots that can store numbers. The directory should store numbers, check if a certain slot is empty or not, and empty a given slot.

Implement the

PhoneDirectory

class:

PhoneDirectory(int maxNumbers)

Initializes the phone directory with the number of available slots

maxNumbers

.

int get()

Provides a number that is not assigned to anyone. Returns

-1

if no number is available.

bool check(int number)

Returns

true

if the slot

number

is available and

false

otherwise.

void release(int number)

Recycles or releases the slot

number

.

Example 1:

Input

["PhoneDirectory", "get", "get", "check", "get", "check", "release", "check"] [[3], [], [], [2], [], [2], [2], [2]]

Output

[null, 0, 1, true, 2, false, null, true]

Explanation

PhoneDirectory phoneDirectory = new PhoneDirectory(3); phoneDirectory.get(); // It can return any available phone number. Here we assume it returns 0. phoneDirectory.get(); // Assume it returns 1. phoneDirectory.check(2); // The number 2 is available, so return true. phoneDirectory.get(); // It returns 2, the only number that is left. phoneDirectory.check(2); // The number 2 is no longer available, so return false. phoneDirectory.release(2); // Release number 2 back to the pool. phoneDirectory.check(2); // Number 2 is available again, return true.

Constraints:

1 <= maxNumbers <= 10

4

0 <= number < maxNumbers

At most

2 * 10

4

calls will be made to

get

,

check

, and

release

.

Code Snippets

C++:

```
class PhoneDirectory {
public:
PhoneDirectory(int maxNumbers) {

}

int get() {

}

bool check(int number) {

}

void release(int number) {

}
};

/**
* Your PhoneDirectory object will be instantiated and called as such:
* PhoneDirectory* obj = new PhoneDirectory(maxNumbers);
* int param_1 = obj->get();
* bool param_2 = obj->check(number);
* obj->release(number);
*/
```

Java:

```
class PhoneDirectory {

public PhoneDirectory(int maxNumbers) {

}

public int get() {

}

public boolean check(int number) {
```

```
}

public void release(int number) {

}
}

/**
* Your PhoneDirectory object will be instantiated and called as such:
* PhoneDirectory obj = new PhoneDirectory(maxNumbers);
* int param_1 = obj.get();
* boolean param_2 = obj.check(number);
* obj.release(number);
*/
```

Python3:

```
class PhoneDirectory:

def __init__(self, maxNumbers: int):


def get(self) -> int:


def check(self, number: int) -> bool:


def release(self, number: int) -> None:



# Your PhoneDirectory object will be instantiated and called as such:
# obj = PhoneDirectory(maxNumbers)
# param_1 = obj.get()
# param_2 = obj.check(number)
# obj.release(number)
```

Python:

```
class PhoneDirectory(object):
```

```
def __init__(self, maxNumbers):
"""
:type maxNumbers: int
"""


def get(self):
"""
:rtype: int
"""


def check(self, number):
"""
:type number: int
:rtype: bool
"""


def release(self, number):
"""
:type number: int
:rtype: None
"""



# Your PhoneDirectory object will be instantiated and called as such:
# obj = PhoneDirectory(maxNumbers)
# param_1 = obj.get()
# param_2 = obj.check(number)
# obj.release(number)
```

JavaScript:

```
/**
* @param {number} maxNumbers
*/
var PhoneDirectory = function(maxNumbers) {

};
```

```
/**
* @return {number}
*/
PhoneDirectory.prototype.get = function() {

};

/**
* @param {number} number
* @return {boolean}
*/
PhoneDirectory.prototype.check = function(number) {

};

/**
* @param {number} number
* @return {void}
*/
PhoneDirectory.prototype.release = function(number) {

};

/**
* Your PhoneDirectory object will be instantiated and called as such:
* var obj = new PhoneDirectory(maxNumbers)
* var param_1 = obj.get()
* var param_2 = obj.check(number)
* obj.release(number)
*/
```

TypeScript:

```
class PhoneDirectory {
constructor(maxNumbers: number) {

}

get(): number {

}
```

```
check(number: number): boolean {

}

release(number: number): void {

}
}

/**
* Your PhoneDirectory object will be instantiated and called as such:
* var obj = new PhoneDirectory(maxNumbers)
* var param_1 = obj.get()
* var param_2 = obj.check(number)
* obj.release(number)
*/
```

C#:

```
public class PhoneDirectory {

public PhoneDirectory(int maxNumbers) {

}

public int Get() {

}

public bool Check(int number) {

}

public void Release(int number) {

}
}

/**
* Your PhoneDirectory object will be instantiated and called as such:
* PhoneDirectory obj = new PhoneDirectory(maxNumbers);
* int param_1 = obj.Get();
```

```
* bool param_2 = obj.Check(number);
* obj.Release(number);
*/
```

C:

```
typedef struct {

} PhoneDirectory;


PhoneDirectory* phoneDirectoryCreate(int maxNumbers) {

}

int phoneDirectoryGet(PhoneDirectory* obj) {

}

bool phoneDirectoryCheck(PhoneDirectory* obj, int number) {

}

void phoneDirectoryRelease(PhoneDirectory* obj, int number) {

}

void phoneDirectoryFree(PhoneDirectory* obj) {

}

/**
* Your PhoneDirectory struct will be instantiated and called as such:
* PhoneDirectory* obj = phoneDirectoryCreate(maxNumbers);
* int param_1 = phoneDirectoryGet(obj);

* bool param_2 = phoneDirectoryCheck(obj, number);

* phoneDirectoryRelease(obj, number);
```

```
* phoneDirectoryFree(obj);
*/
```

Go:

```
type PhoneDirectory struct {

}


func Constructor(maxNumbers int) PhoneDirectory {

}


func (this *PhoneDirectory) Get() int {

}


func (this *PhoneDirectory) Check(number int) bool {

}


func (this *PhoneDirectory) Release(number int) {

}


/**
* Your PhoneDirectory object will be instantiated and called as such:
* obj := Constructor(maxNumbers);
* param_1 := obj.Get();
* param_2 := obj.Check(number);
* obj.Release(number);
*/
```

Kotlin:

```
class PhoneDirectory(maxNumbers: Int) {

fun get(): Int {

}

fun check(number: Int): Boolean {

}

fun release(number: Int) {

}

}

/**
* Your PhoneDirectory object will be instantiated and called as such:
* var obj = PhoneDirectory(maxNumbers)
* var param_1 = obj.get()
* var param_2 = obj.check(number)
* obj.release(number)
*/
```

Swift:

```
class PhoneDirectory {

init(_ maxNumbers: Int) {

}

func get() -> Int {

}

func check(_ number: Int) -> Bool {

}

func release(_ number: Int) {
```

```
}
}

/**
* Your PhoneDirectory object will be instantiated and called as such:
* let obj = PhoneDirectory(maxNumbers)
* let ret_1: Int = obj.get()
* let ret_2: Bool = obj.check(number)
* obj.release(number)
*/
```

Rust:

```
struct PhoneDirectory {

}


/**
* `&self` means the method takes an immutable reference.
* If you need a mutable reference, change it to `&mut self` instead.
*/
impl PhoneDirectory {

fn new(maxNumbers: i32) -> Self {

}

fn get(&self) -> i32 {

}

fn check(&self, number: i32) -> bool {

}

fn release(&self, number: i32) {

}
}

/**
```

```
* Your PhoneDirectory object will be instantiated and called as such:
* let obj = PhoneDirectory::new(maxNumbers);
* let ret_1: i32 = obj.get();
* let ret_2: bool = obj.check(number);
* obj.release(number);
*/
```

Ruby:

```
class PhoneDirectory

=begin
:type max_numbers: Integer
=end
def initialize(max_numbers)

end


=begin
:rtype: Integer
=end
def get()

end


=begin
:type number: Integer
:rtype: Boolean
=end
def check(number)

end


=begin
:type number: Integer
:rtype: Void
=end
def release(number)
```

```
end


end

# Your PhoneDirectory object will be instantiated and called as such:
# obj = PhoneDirectory.new(max_numbers)
# param_1 = obj.get()
# param_2 = obj.check(number)
# obj.release(number)
```

PHP:

```
class PhoneDirectory {
/**
* @param Integer $maxNumbers
*/
function __construct($maxNumbers) {

}

/**
* @return Integer
*/
function get() {

}

/**
* @param Integer $number
* @return Boolean
*/
function check($number) {

}

/**
* @param Integer $number
* @return NULL
*/
function release($number) {
```

```
}
}

/**
* Your PhoneDirectory object will be instantiated and called as such:
* $obj = PhoneDirectory($maxNumbers);
* $ret_1 = $obj->get();
* $ret_2 = $obj->check($number);
* $obj->release($number);
*/
```

Dart:

```
class PhoneDirectory {

PhoneDirectory(int maxNumbers) {

}

int get() {

}

bool check(int number) {

}

void release(int number) {

}
}

/**
* Your PhoneDirectory object will be instantiated and called as such:
* PhoneDirectory obj = PhoneDirectory(maxNumbers);
* int param1 = obj.get();
* bool param2 = obj.check(number);
* obj.release(number);
*/
```

Scala:

```
class PhoneDirectory(_maxNumbers: Int) {

def get(): Int = {

}

def check(number: Int): Boolean = {

}

def release(number: Int): Unit = {

}

}

/**
* Your PhoneDirectory object will be instantiated and called as such:
* val obj = new PhoneDirectory(maxNumbers)
* val param_1 = obj.get()
* val param_2 = obj.check(number)
* obj.release(number)
*/
```

Elixir:

```
defmodule PhoneDirectory do
@spec init_(max_numbers :: integer) :: any
def init_(max_numbers) do

end

@spec get() :: integer
def get() do

end

@spec check(number :: integer) :: boolean
def check(number) do

end

@spec release(number :: integer) :: any
```

```
def release(number) do

end
end

# Your functions will be called as such:
# PhoneDirectory.init_(max_numbers)
# param_1 = PhoneDirectory.get()
# param_2 = PhoneDirectory.check(number)
# PhoneDirectory.release(number)

# PhoneDirectory.init_ will be called before every test case, in which you
can do some necessary initializations.
```

Erlang:

```
-spec phone_directory_init_(MaxNumbers :: integer()) -> any().
phone_directory_init_(MaxNumbers) ->
.

-spec phone_directory_get() -> integer().
phone_directory_get() ->
.

-spec phone_directory_check(Number :: integer()) -> boolean().
phone_directory_check(Number) ->
.

-spec phone_directory_release(Number :: integer()) -> any().
phone_directory_release(Number) ->
.


%% Your functions will be called as such:
%% phone_directory_init_(MaxNumbers),
%% Param_1 = phone_directory_get(),
%% Param_2 = phone_directory_check(Number),
%% phone_directory_release(Number),

%% phone_directory_init_ will be called before every test case, in which you
can do some necessary initializations.
```

Racket:

```
(define phone-directory%
(class object%
(super-new)

; max-numbers : exact-integer?
(init-field
max-numbers)

; get : -> exact-integer?
(define/public (get)
)
; check : exact-integer? -> boolean?
(define/public (check number)
)
; release : exact-integer? -> void?
(define/public (release number)
)))

;; Your phone-directory% object will be instantiated and called as such:
;; (define obj (new phone-directory% [max-numbers max-numbers]))
;; (define param_1 (send obj get))
;; (define param_2 (send obj check number))
;; (send obj release number)
```

Solutions

C++ Solution:

```
/*
* Problem: Design Phone Directory
* Difficulty: Medium
* Tags: array, hash, linked_list, queue
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class PhoneDirectory {
```

```
public:
PhoneDirectory(int maxNumbers) {

}

int get() {

}

bool check(int number) {

}

void release(int number) {

}
};

/**
* Your PhoneDirectory object will be instantiated and called as such:
* PhoneDirectory* obj = new PhoneDirectory(maxNumbers);
* int param_1 = obj->get();
* bool param_2 = obj->check(number);
* obj->release(number);
*/
```

Java Solution:

```
/**
* Problem: Design Phone Directory
* Difficulty: Medium
* Tags: array, hash, linked_list, queue
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class PhoneDirectory {

public PhoneDirectory(int maxNumbers) {
```

```
}

public int get() {

}

public boolean check(int number) {

}

public void release(int number) {

}
}

/**
* Your PhoneDirectory object will be instantiated and called as such:
* PhoneDirectory obj = new PhoneDirectory(maxNumbers);
* int param_1 = obj.get();
* boolean param_2 = obj.check(number);
* obj.release(number);
*/
```

Python3 Solution:

```
"""
Problem: Design Phone Directory
Difficulty: Medium
Tags: array, hash, linked_list, queue

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

class PhoneDirectory:

def __init__(self, maxNumbers: int):
```

```
def get(self) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class PhoneDirectory(object):

def __init__(self, maxNumbers):
"""
:type maxNumbers: int
"""



def get(self):
"""
:rtype: int
"""



def check(self, number):
"""
:type number: int
:rtype: bool
"""



def release(self, number):
"""
:type number: int
:rtype: None
"""




# Your PhoneDirectory object will be instantiated and called as such:
# obj = PhoneDirectory(maxNumbers)
# param_1 = obj.get()
# param_2 = obj.check(number)
# obj.release(number)
```

JavaScript Solution:

```
/**
* Problem: Design Phone Directory
* Difficulty: Medium
* Tags: array, hash, linked_list, queue
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

/**
* @param {number} maxNumbers
*/
var PhoneDirectory = function(maxNumbers) {

};

/**
* @return {number}
*/
PhoneDirectory.prototype.get = function() {

};

/**
* @param {number} number
* @return {boolean}
*/
PhoneDirectory.prototype.check = function(number) {

};

/**
* @param {number} number
* @return {void}
*/
PhoneDirectory.prototype.release = function(number) {

};

/**
```

```
* Your PhoneDirectory object will be instantiated and called as such:
* var obj = new PhoneDirectory(maxNumbers)
* var param_1 = obj.get()
* var param_2 = obj.check(number)
* obj.release(number)
*/
```

TypeScript Solution:

```
/**
* Problem: Design Phone Directory
* Difficulty: Medium
* Tags: array, hash, linked_list, queue
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class PhoneDirectory {
constructor(maxNumbers: number) {

}

get(): number {

}

check(number: number): boolean {

}

release(number: number): void {

}
}

/**
* Your PhoneDirectory object will be instantiated and called as such:
* var obj = new PhoneDirectory(maxNumbers)
* var param_1 = obj.get()
```

```
* var param_2 = obj.check(number)
* obj.release(number)
*/
```

C# Solution:

```
/*
* Problem: Design Phone Directory
* Difficulty: Medium
* Tags: array, hash, linked_list, queue
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class PhoneDirectory {

public PhoneDirectory(int maxNumbers) {

}

public int Get() {

}

public bool Check(int number) {

}

public void Release(int number) {

}
}

/**
* Your PhoneDirectory object will be instantiated and called as such:
* PhoneDirectory obj = new PhoneDirectory(maxNumbers);
* int param_1 = obj.Get();
* bool param_2 = obj.Check(number);
* obj.Release(number);
```

```
*/
```

C Solution:

```
/*
* Problem: Design Phone Directory
* Difficulty: Medium
* Tags: array, hash, linked_list, queue
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/




typedef struct {

} PhoneDirectory;


PhoneDirectory* phoneDirectoryCreate(int maxNumbers) {

}

int phoneDirectoryGet(PhoneDirectory* obj) {

}

bool phoneDirectoryCheck(PhoneDirectory* obj, int number) {

}

void phoneDirectoryRelease(PhoneDirectory* obj, int number) {

}

void phoneDirectoryFree(PhoneDirectory* obj) {

}
```

```
/**
* Your PhoneDirectory struct will be instantiated and called as such:
* PhoneDirectory* obj = phoneDirectoryCreate(maxNumbers);
* int param_1 = phoneDirectoryGet(obj);

* bool param_2 = phoneDirectoryCheck(obj, number);

* phoneDirectoryRelease(obj, number);

* phoneDirectoryFree(obj);
*/
```

Go Solution:

```
// Problem: Design Phone Directory
// Difficulty: Medium
// Tags: array, hash, linked_list, queue
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

type PhoneDirectory struct {

}


func Constructor(maxNumbers int) PhoneDirectory {

}


func (this *PhoneDirectory) Get() int {

}


func (this *PhoneDirectory) Check(number int) bool {

}
```

```
func (this *PhoneDirectory) Release(number int) {

}


/**
* Your PhoneDirectory object will be instantiated and called as such:
* obj := Constructor(maxNumbers);
* param_1 := obj.Get();
* param_2 := obj.Check(number);
* obj.Release(number);
*/
```

Kotlin Solution:

```
class PhoneDirectory(maxNumbers: Int) {

fun get(): Int {

}

fun check(number: Int): Boolean {

}

fun release(number: Int) {

}

}

/**
* Your PhoneDirectory object will be instantiated and called as such:
* var obj = PhoneDirectory(maxNumbers)
* var param_1 = obj.get()
* var param_2 = obj.check(number)
* obj.release(number)
*/
```

Swift Solution:

```
class PhoneDirectory {

init(_ maxNumbers: Int) {

}

func get() -> Int {

}

func check(_ number: Int) -> Bool {

}

func release(_ number: Int) {

}
}

/**
* Your PhoneDirectory object will be instantiated and called as such:
* let obj = PhoneDirectory(maxNumbers)
* let ret_1: Int = obj.get()
* let ret_2: Bool = obj.check(number)
* obj.release(number)
*/
```

Rust Solution:

```
// Problem: Design Phone Directory
// Difficulty: Medium
// Tags: array, hash, linked_list, queue
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

struct PhoneDirectory {

}
```

```
/**
* `&self` means the method takes an immutable reference.
* If you need a mutable reference, change it to `&mut self` instead.
*/
impl PhoneDirectory {

fn new(maxNumbers: i32) -> Self {

}

fn get(&self) -> i32 {

}

fn check(&self, number: i32) -> bool {

}

fn release(&self, number: i32) {

}
}

/**
* Your PhoneDirectory object will be instantiated and called as such:
* let obj = PhoneDirectory::new(maxNumbers);
* let ret_1: i32 = obj.get();
* let ret_2: bool = obj.check(number);
* obj.release(number);
*/
```

Ruby Solution:

```
class PhoneDirectory

=begin
:type max_numbers: Integer
=end
def initialize(max_numbers)
```

```
end


=begin
:rtype: Integer
=end
def get()

end


=begin
:type number: Integer
:rtype: Boolean
=end
def check(number)

end


=begin
:type number: Integer
:rtype: Void
=end
def release(number)

end


end

# Your PhoneDirectory object will be instantiated and called as such:
# obj = PhoneDirectory.new(max_numbers)
# param_1 = obj.get()
# param_2 = obj.check(number)
# obj.release(number)
```

PHP Solution:

```
class PhoneDirectory {
/**
* @param Integer $maxNumbers
*/
function __construct($maxNumbers) {

}

/**
* @return Integer
*/
function get() {

}

/**
* @param Integer $number
* @return Boolean
*/
function check($number) {

}

/**
* @param Integer $number
* @return NULL
*/
function release($number) {

}
}

/**
* Your PhoneDirectory object will be instantiated and called as such:
* $obj = PhoneDirectory($maxNumbers);
* $ret_1 = $obj->get();
* $ret_2 = $obj->check($number);
* $obj->release($number);
*/
```

Dart Solution:

```
class PhoneDirectory {

PhoneDirectory(int maxNumbers) {

}

int get() {

}

bool check(int number) {

}

void release(int number) {

}
}

/**
* Your PhoneDirectory object will be instantiated and called as such:
* PhoneDirectory obj = PhoneDirectory(maxNumbers);
* int param1 = obj.get();
* bool param2 = obj.check(number);
* obj.release(number);
*/
```

Scala Solution:

```
class PhoneDirectory(_maxNumbers: Int) {

def get(): Int = {

}

def check(number: Int): Boolean = {

}

def release(number: Int): Unit = {

}
```

```
}

/**
* Your PhoneDirectory object will be instantiated and called as such:
* val obj = new PhoneDirectory(maxNumbers)
* val param_1 = obj.get()
* val param_2 = obj.check(number)
* obj.release(number)
*/
```

Elixir Solution:

```
defmodule PhoneDirectory do
@spec init_(max_numbers :: integer) :: any
def init_(max_numbers) do

end

@spec get() :: integer
def get() do

end

@spec check(number :: integer) :: boolean
def check(number) do

end

@spec release(number :: integer) :: any
def release(number) do

end
end

# Your functions will be called as such:
# PhoneDirectory.init_(max_numbers)
# param_1 = PhoneDirectory.get()
# param_2 = PhoneDirectory.check(number)
# PhoneDirectory.release(number)
```

```
# PhoneDirectory.init_ will be called before every test case, in which you
can do some necessary initializations.
```

Erlang Solution:

```
-spec phone_directory_init_(MaxNumbers :: integer()) -> any().
phone_directory_init_(MaxNumbers) ->
.

-spec phone_directory_get() -> integer().
phone_directory_get() ->
.

-spec phone_directory_check(Number :: integer()) -> boolean().
phone_directory_check(Number) ->
.

-spec phone_directory_release(Number :: integer()) -> any().
phone_directory_release(Number) ->
.


%% Your functions will be called as such:
%% phone_directory_init_(MaxNumbers),
%% Param_1 = phone_directory_get(),
%% Param_2 = phone_directory_check(Number),
%% phone_directory_release(Number),

%% phone_directory_init_ will be called before every test case, in which you
can do some necessary initializations.
```

Racket Solution:

```
(define phone-directory%
(class object%
(super-new)

; max-numbers : exact-integer?
(init-field
max-numbers)
```

```
; get : -> exact-integer?
(define/public (get)
)
; check : exact-integer? -> boolean?
(define/public (check number)
)
; release : exact-integer? -> void?
(define/public (release number)
)))

;; Your phone-directory% object will be instantiated and called as such:
;; (define obj (new phone-directory% [max-numbers max-numbers]))
;; (define param_1 (send obj get))
;; (define param_2 (send obj check number))
;; (send obj release number)
```