Problem 157: Read N Characters Given Read4

Problem Information

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

Problem Description

Given a

file

and assume that you can only read the file using a given method

read4

, implement a method to read

n

characters.

Method read4:

The API

read4

reads

four consecutive characters

from

file

, then writes those characters into the buffer array

buf4

.

The return value is the number of actual characters read.

Note that

read4()

has its own file pointer, much like

FILE *fp

in C.

Definition of read4:

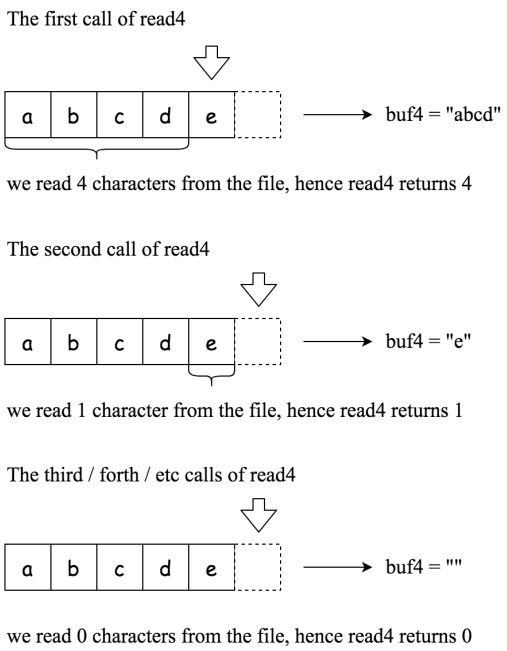
Parameter: char[] buf4 Returns: int

buf4[] is a destination, not a source. The results from read4 will be copied to buf4[].

Below is a high-level example of how

read4

works:



File file("abcde

"); // File is "

abcde

", initially file pointer (fp) points to 'a' char[] buf4 = new char[4]; // Create buffer with enough space to store characters read4(buf4); // read4 returns 4. Now buf4 = "abcd", fp points to 'e' read4(buf4); // read4 returns 1. Now buf4 = "e", fp points to end of file read4(buf4); // read4 returns 0. Now buf4 = "", fp points to end of file

Method read:

By using the

read4

method, implement the method read that reads

n

characters from

file

and store it in the buffer array

buf

. Consider that you cannot manipulate

file

directly.

The return value is the number of actual characters read.

Definition of read:

Parameters: char[] buf, int n Returns: int

buf[] is a destination, not a source. You will need to write the results to buf[].

Note:

Consider that you cannot manipulate the file directly. The file is only accessible for

read4

but not for

read

.

The

read

function will only be called once for each test case.

You may assume the destination buffer array,

buf

, is guaranteed to have enough space for storing

n

characters.

Example 1:

Input:

file = "abc", n = 4

Output:

3

Explanation:

After calling your read method, buf should contain "abc". We read a total of 3 characters from the file, so return 3. Note that "abc" is the file's content, not buf. buf is the destination buffer that you will have to write the results to.

Example 2:

Input:

file = "abcde", n = 5

Output:

5

Explanation:

After calling your read method, buf should contain "abcde". We read a total of 5 characters from the file, so return 5.

Example 3:

Input:

file = "abcdABCD1234", n = 12

Output:

12

Explanation:

After calling your read method, buf should contain "abcdABCD1234". We read a total of 12 characters from the file, so return 12.

Constraints:

1 <= file.length <= 500

file

consist of English letters and digits.

1 <= n <= 1000

Code Snippets

C++:

```
/**
* The read4 API is defined in the parent class Reader4.
* int read4(char *buf4);
*/

class Solution {
public:
/**
* @param buf Destination buffer
* @param n Number of characters to read
```

```
* @return The number of actual characters read
*/
int read(char *buf, int n) {

}
};
```

Java:

```
/**
* The read4 API is defined in the parent class Reader4.
* int read4(char[] buf4);
*/

public class Solution extends Reader4 {
/**
* @param buf Destination buffer
* @param n Number of characters to read
* @return The number of actual characters read
*/
public int read(char[] buf, int n) {

}
}
```

Python3:

```
"""
The read4 API is already defined for you.

@param buf4, a list of characters
@return an integer
def read4(buf4):

# Below is an example of how the read4 API can be called.
file = File("abcdefghijk") # File is "abcdefghijk", initially file pointer
(fp) points to 'a'
buf4 = [' '] * 4 # Create buffer with enough space to store characters
read4(buf4) # read4 returns 4. Now buf = ['a','b','c','d'], fp points to 'e'
read4(buf4) # read4 returns 4. Now buf = ['e','f','g','h'], fp points to 'i'
read4(buf4) # read4 returns 3. Now buf = ['i','j','k',...], fp points to end
of file
```

```
"""

class Solution:
def read(self, buf, n):
"""
:type buf: Destination buffer (List[str])
:type n: Number of characters to read (int)
:rtype: The number of actual characters read (int)
"""
```

Python:

```
"""
The read4 API is already defined for you.

@param buf4, a list of characters
@return an integer
def read4(buf4):

# Below is an example of how the read4 API can be called.
file = File("abcdefghijk") # File is "abcdefghijk", initially file pointer
(fp) points to 'a'
buf4 = [' '] * 4 # Create buffer with enough space to store characters
read4(buf4) # read4 returns 4. Now buf = ['a','b','c','d'], fp points to 'e'
read4(buf4) # read4 returns 4. Now buf = ['e','f','g','h'], fp points to 'i'
read4(buf4) # read4 returns 3. Now buf = ['i','j','k',...], fp points to end
of file
"""

class Solution(object):
def read(self, buf, n):
"""
:type buf: Destination buffer (List[str])
:type n: Number of characters to read (int)
:rtype: The number of actual characters read (int)
"""
```

JavaScript:

```
/**
* Definition for read4()
*
```

```
* @param {character[]} buf4 Destination buffer
* @return {number} The number of actual characters read
* read4 = function(buf4) {
* ...
* };
*/

/**
* @param {function} read4()
* @return {function}
*/
var solution = function(read4) {
/**
* @param {character[]} buf Destination buffer
* @param {number} n Number of characters to read
* @return {number} The number of actual characters read
*/
return function(buf, n) {

};
};
```

TypeScript:

```
/**
* Definition for read4()
* read4 = function(buf4: string[]): number {
* ...
* };
*/

var solution = function(read4: any) {

return function(buf: string[], n: number): number {

};
};
```

C#:

```
/**
* The Read4 API is defined in the parent class Reader4.
```

```
* int Read4(char[] buf4);
*/

public class Solution : Reader4 {
/**
* @param buf Destination buffer
* @param n Number of characters to read
* @return The number of actual characters read
*/
public int Read(char[] buf, int n) {

}
}
```

C:

```
/**
* The read4 API is defined in the parent class Reader4.
* int read4(char *buf4);
*/

/**
* @param buf Destination buffer
* @param n Number of characters to read
* @return The number of actual characters read
*/
int _read(char* buf, int n) {

}
```

Go:

```
/**
* The read4 API is already defined for you.
*
* read4 := func(buf4 []byte) int
*
* // Below is an example of how the read4 API can be called.
* file := File("abcdefghijk") // File is "abcdefghijk", initially file
pointer (fp) points to 'a'
* buf4 := make([]byte, 4) // Create buffer with enough space to store
characters
```

```
* read4(buf4) // read4 returns 4. Now buf = ['a','b','c','d'], fp points to
'e'
* read4(buf4) // read4 returns 4. Now buf = ['e','f','g','h'], fp points to
'i'
* read4(buf4) // read4 returns 3. Now buf = ['i','j','k',...], fp points to
end of file
*/

var solution = func(read4 func([]byte) int) func([]byte, int) int {
// implement read below.
return func(buf []byte, n int) int {

}
}
```

Kotlin:

```
/**
* The read4 API is defined in the parent class Reader4.
* fun read4(buf4:CharArray): Int {}
*/

class Solution:Reader4() {
/**
* @param buf Destination buffer
* @param n Number of characters to read
* @return The number of actual characters read
*/
override fun read(buf:CharArray, n:Int): Int {

}
}
```

Swift:

```
/**
* The read4 API is defined in the parent class Reader4.
* func read4(_ buf4: inout [Character]) -> Int;
*/

class Solution : Reader4 {
/**
```

```
* @param buf Destination buffer
* @param n Number of characters to read
* @return The number of actual characters read
*/
func read(_ buf: inout [Character], _ n: Int) -> Int {

}
}
```

Rust:

```
/**
* The read4 API is defined as.
* fn read4(&self,buf4: &mut [char]) -> i32;
* You can call it using self.read4(buf4)
*/

impl Solution {
pub fn read(&self, buf: &mut [char], n: i32) -> i32 {

}
}
```

Ruby:

```
# The read4 API is already defined for you.
# Below is an example of how the read4 API can be called.
# file = File.new("abcdefghijk") File is "abcdefghijk", initially file
pointer (fp) points to 'a'
# buf4 = [' '] * 4 Create buffer with enough space to store characters
# read4(buf4) # read4 returns 4. Now buf = ['a','b','c','d'], fp points to
'e'
# read4(buf4) # read4 returns 4. Now buf = ['e','f','g','h'], fp points to
'i'
# read4(buf4) # read4 returns 3. Now buf = ['i','j','k',...], fp points to
end of file

# @param {List[str]} buf
# @param {int} n
# @return {int}
def read(buf, n)
```

```
end
```

PHP:

```
/* The read4 API is defined in the parent class Reader4.
public function read4(&$buf4){} */

class Solution extends Reader4 {
/**
* @param Char[] &$buf Destination buffer
* @param Integer $n Number of characters to read
* @return Integer The number of actual characters read
*/
function read(&$buf, $n) {

}
}
```

Scala:

```
/**
* The read4 API is defined in the parent class Reader4.
* def read4(buf4: Array[Char]): Int = {}
*/

class Solution extends Reader4 {
/**
* @param buf Destination buffer
* @param n Number of characters to read
* @return The number of actual characters read
*/
def read(buf: Array[Char], n: Int): Int = {

}
}
```

Solutions

C++ Solution:

```
/*
* Problem: Read N Characters Given Read4
* Difficulty: Easy
* Tags: array
*
* 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
*/

/**
* The read4 API is defined in the parent class Reader4.
* int read4(char *buf4);
*/

class Solution {
public:
/**
* @param buf Destination buffer
* @param n Number of characters to read
* @return The number of actual characters read
*/
int read(char *buf, int n) {

}
};
```

Java Solution:

```
/**
* Problem: Read N Characters Given Read4
* Difficulty: Easy
* Tags: array
*
* 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
*/

/**
* The read4 API is defined in the parent class Reader4.
* int read4(char[] buf4);
```

```
*/

public class Solution extends Reader4 {
/**
* @param buf Destination buffer
* @param n Number of characters to read
* @return The number of actual characters read
*/
public int read(char[] buf, int n) {

}
}
```

Python3 Solution:

```
"""
The read4 API is already defined for you.

@param buf4, a list of characters
@return an integer
def read4(buf4):

# Below is an example of how the read4 API can be called.
file = File("abcdefghijk") # File is "abcdefghijk", initially file pointer
(fp) points to 'a'
buf4 = [' '] * 4 # Create buffer with enough space to store characters
read4(buf4) # read4 returns 4. Now buf = ['a','b','c','d'], fp points to 'e'
read4(buf4) # read4 returns 4. Now buf = ['e','f','g','h'], fp points to 'i'
read4(buf4) # read4 returns 3. Now buf = ['i','j','k',...], fp points to end
of file
"""

class Solution:
def read(self, buf, n):
"""
:type buf: Destination buffer (List[str])
:type n: Number of characters to read (int)
:rtype: The number of actual characters read (int)
"""
```

Python Solution:

```
"""
The read4 API is already defined for you.

@param buf4, a list of characters
@return an integer
def read4(buf4):

# Below is an example of how the read4 API can be called.
file = File("abcdefghijk") # File is "abcdefghijk", initially file pointer
(fp) points to 'a'
buf4 = [' '] * 4 # Create buffer with enough space to store characters
read4(buf4) # read4 returns 4. Now buf = ['a','b','c','d'], fp points to 'e'
read4(buf4) # read4 returns 4. Now buf = ['e','f','g','h'], fp points to 'i'
read4(buf4) # read4 returns 3. Now buf = ['i','j','k',...], fp points to end
of file
"""

class Solution(object):
def read(self, buf, n):
"""
:type buf: Destination buffer (List[str])
:type n: Number of characters to read (int)
:rtype: The number of actual characters read (int)
"""
```

JavaScript Solution:

```
/**
* Problem: Read N Characters Given Read4
* Difficulty: Easy
* Tags: array
*
* 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
*/

/**
* Definition for read4()
*
* @param {character[]} buf4 Destination buffer
* @return {number} The number of actual characters read
```

```
* read4 = function(buf4) {
* ...
* };
*/

/**
* @param {function} read4()
* @return {function}
*/
var solution = function(read4) {
/**
* @param {character[]} buf Destination buffer
* @param {number} n Number of characters to read
* @return {number} The number of actual characters read
*/
return function(buf, n) {

};
};
```

TypeScript Solution:

```
/**
* Problem: Read N Characters Given Read4
* Difficulty: Easy
* Tags: array
*
* 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
*/

/**
* Definition for read4()
* read4 = function(buf4: string[]): number {
* ...
* };
*/

var solution = function(read4: any) {
```

```
return function(buf: string[], n: number): number {

};
};
```

C# Solution:

```
/*
* Problem: Read N Characters Given Read4
* Difficulty: Easy
* Tags: array
*
* 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
*/

/**
* The Read4 API is defined in the parent class Reader4.
* int Read4(char[] buf4);
*/

public class Solution : Reader4 {
/**
* @param buf Destination buffer
* @param n Number of characters to read
* @return The number of actual characters read
*/
public int Read(char[] buf, int n) {

}
}
```

C Solution:

```
/*
* Problem: Read N Characters Given Read4
* Difficulty: Easy
* Tags: array
*
* 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
*/

/**
* The read4 API is defined in the parent class Reader4.
* int read4(char *buf4);
*/

/**
* @param buf Destination buffer
* @param n Number of characters to read
* @return The number of actual characters read
*/
int _read(char* buf, int n) {

}
```

Go Solution:

```
// Problem: Read N Characters Given Read4
// Difficulty: Easy
// Tags: array
//
// 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

/**
* The read4 API is already defined for you.
*
* read4 := func(buf4 []byte) int
*
* // Below is an example of how the read4 API can be called.
* file := File("abcdefghijk") // File is "abcdefghijk", initially file
pointer (fp) points to 'a'
* buf4 := make([]byte, 4) // Create buffer with enough space to store
characters
* read4(buf4) // read4 returns 4. Now buf = ['a','b','c','d'], fp points to
'e'
* read4(buf4) // read4 returns 4. Now buf = ['e','f','g','h'], fp points to
```

```
'i'
* read4(buf4) // read4 returns 3. Now buf = ['i','j','k',...], fp points to
end of file
*/

var solution = func(read4 func([]byte) int) func([]byte, int) int {
// implement read below.
return func(buf []byte, n int) int {

}
}
```

Kotlin Solution:

```
/**
* The read4 API is defined in the parent class Reader4.
* fun read4(buf4:CharArray): Int {}
*/

class Solution:Reader4() {
/**
* @param buf Destination buffer
* @param n Number of characters to read
* @return The number of actual characters read
*/
override fun read(buf:CharArray, n:Int): Int {

}
}
```

Swift Solution:

```
/**
* The read4 API is defined in the parent class Reader4.
* func read4(_ buf4: inout [Character]) -> Int;
*/

class Solution : Reader4 {
/**
* @param buf Destination buffer
* @param n Number of characters to read
* @return The number of actual characters read
```

```
*/
func read(_ buf: inout [Character], _ n: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Read N Characters Given Read4
// Difficulty: Easy
// Tags: array
//
// 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

/**
* The read4 API is defined as.
* fn read4(&self,buf4: &mut [char]) -> i32;
* You can call it using self.read4(buf4)
*/

impl Solution {
pub fn read(&self, buf: &mut [char], n: i32) -> i32 {

}
}
```

Ruby Solution:

```
# The read4 API is already defined for you.
# Below is an example of how the read4 API can be called.
# file = File.new("abcdefghijk") File is "abcdefghijk", initially file
pointer (fp) points to 'a'
# buf4 = [' '] * 4 Create buffer with enough space to store characters
# read4(buf4) # read4 returns 4. Now buf = ['a','b','c','d'], fp points to
'e'
# read4(buf4) # read4 returns 4. Now buf = ['e','f','g','h'], fp points to
'i'
# read4(buf4) # read4 returns 3. Now buf = ['i','j','k',...], fp points to
end of file
```

```
# @param {List[str]} buf
# @param {int} n
# @return {int}
def read(buf, n)

end
```

PHP Solution:

```
/* The read4 API is defined in the parent class Reader4.
public function read4(&$buf4){} */

class Solution extends Reader4 {
/**
* @param Char[] &$buf Destination buffer
* @param Integer $n Number of characters to read
* @return Integer The number of actual characters read
*/
function read(&$buf, $n) {

}
}
```

Scala Solution:

```
/**
* The read4 API is defined in the parent class Reader4.
* def read4(buf4: Array[Char]): Int = {}
*/

class Solution extends Reader4 {
/**
* @param buf Destination buffer
* @param n Number of characters to read
* @return The number of actual characters read
*/
def read(buf: Array[Char], n: Int): Int = {

}
}
```