Problem 880: Decoded String at Index

Problem Information

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

Problem Description

You are given an encoded string

s

. To decode the string to a tape, the encoded string is read one character at a time and the following steps are taken:

If the character read is a letter, that letter is written onto the tape.

If the character read is a digit

d

, the entire current tape is repeatedly written

d - 1

more times in total.

Given an integer

k

, return

the

k

th

letter (

1-indexed)

in the decoded string

.

Example 1:

Input:

s = "leet2code3", k = 10

Output:

"o"

Explanation:

The decoded string is "leetleetcodeleetleetcodeleetleetcode". The 10

th

letter in the string is "o".

Example 2:

Input:

s = "ha22", k = 5

Output:

"h"

Explanation:

The decoded string is "hahahaha". The 5

th

letter is "h".

Example 3:

Input:

s = "a2345678999999999999999", k = 1

Output:

"a"

Explanation:

The decoded string is "a" repeated 8301530446056247680 times. The 1

st

letter is "a".

Constraints:

2 <= s.length <= 100

s

consists of lowercase English letters and digits

2

through

9

.

s

starts with a letter.

1 <= k <= 10

9

It is guaranteed that

k

is less than or equal to the length of the decoded string.

The decoded string is guaranteed to have less than

2

63

letters.

Code Snippets

C++:

```
class Solution {
public:
string decodeAtIndex(string s, int k) {

}
};
```

Java:

```
class Solution {
public String decodeAtIndex(String s, int k) {

}
}
```

Python3:

```
class Solution:
def decodeAtIndex(self, s: str, k: int) -> str:
```

Python:

```
class Solution(object):
def decodeAtIndex(self, s, k):
"""
:type s: str
:type k: int
:rtype: str
"""
```

JavaScript:

```
/**
* @param {string} s
* @param {number} k
* @return {string}
*/
var decodeAtIndex = function(s, k) {

};
```

TypeScript:

```
function decodeAtIndex(s: string, k: number): string {

};
```

C#:

```
public class Solution {
public string DecodeAtIndex(string s, int k) {
}
}
```

C:

```
char* decodeAtIndex(char* s, int k) {

}
```

Go:

```
func decodeAtIndex(s string, k int) string {

}
```

Kotlin:

```
class Solution {
fun decodeAtIndex(s: String, k: Int): String {

}
}
```

Swift:

```
class Solution {
func decodeAtIndex(_ s: String, _ k: Int) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn decode_at_index(s: String, k: i32) -> String {

}
}
```

Ruby:

```
# @param {String} s
# @param {Integer} k
# @return {String}
def decode_at_index(s, k)

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @param Integer $k
* @return String
*/
function decodeAtIndex($s, $k) {

}
}
```

Dart:

```
class Solution {
String decodeAtIndex(String s, int k) {

}
}
```

Scala:

```
object Solution {
def decodeAtIndex(s: String, k: Int): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec decode_at_index(s :: String.t, k :: integer) :: String.t
def decode_at_index(s, k) do
```

```
end
end
```

Erlang:

```
-spec decode_at_index(S :: unicode:unicode_binary(), K :: integer()) ->
unicode:unicode_binary().
decode_at_index(S, K) ->
.
```

Racket:

```
(define/contract (decode-at-index s k)
(-> string? exact-integer? string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Decoded String at Index
* Difficulty: Medium
* Tags: string, stack
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
string decodeAtIndex(string s, int k) {

}
};
```

Java Solution:

```
/**
* Problem: Decoded String at Index
* Difficulty: Medium
* Tags: string, stack
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public String decodeAtIndex(String s, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Decoded String at Index
Difficulty: Medium
Tags: string, stack

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def decodeAtIndex(self, s: str, k: int) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def decodeAtIndex(self, s, k):
"""
:type s: str
:type k: int
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Decoded String at Index
* Difficulty: Medium
* Tags: string, stack
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {string} s
* @param {number} k
* @return {string}
*/
var decodeAtIndex = function(s, k) {

};
```

TypeScript Solution:

```
/**
* Problem: Decoded String at Index
* Difficulty: Medium
* Tags: string, stack
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function decodeAtIndex(s: string, k: number): string {

};
```

C# Solution:

```
/*
* Problem: Decoded String at Index
* Difficulty: Medium
* Tags: string, stack
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public string DecodeAtIndex(string s, int k) {

}
}
```

C Solution:

```
/*
* Problem: Decoded String at Index
* Difficulty: Medium
* Tags: string, stack
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

char* decodeAtIndex(char* s, int k) {

}
```

Go Solution:

```
// Problem: Decoded String at Index
// Difficulty: Medium
// Tags: string, stack
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func decodeAtIndex(s string, k int) string {
}
```

Kotlin Solution:

```
class Solution {
fun decodeAtIndex(s: String, k: Int): String {

}
}
```

Swift Solution:

```
class Solution {
func decodeAtIndex(_ s: String, _ k: Int) -> String {

}
}
```

Rust Solution:

```
// Problem: Decoded String at Index
// Difficulty: Medium
// Tags: string, stack
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn decode_at_index(s: String, k: i32) -> String {

}
}
```

Ruby Solution:

```
# @param {String} s
# @param {Integer} k
# @return {String}
def decode_at_index(s, k)
```

```
end
```

PHP Solution:

```
class Solution {

/**
* @param String $s
* @param Integer $k
* @return String
*/
function decodeAtIndex($s, $k) {

}
}
```

Dart Solution:

```
class Solution {
String decodeAtIndex(String s, int k) {

}
}
```

Scala Solution:

```
object Solution {
def decodeAtIndex(s: String, k: Int): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec decode_at_index(s :: String.t, k :: integer) :: String.t
def decode_at_index(s, k) do

end
end
```

Erlang Solution:

```
-spec decode_at_index(S :: unicode:unicode_binary(), K :: integer()) ->
unicode:unicode_binary().
decode_at_index(S, K) ->
.
```

Racket Solution:

```
(define/contract (decode-at-index s k)
(-> string? exact-integer? string?)
)
```