Problem 604: Design Compressed String Iterator

Problem Information

Difficulty: Easy
Acceptance Rate: 40.21%
Paid Only: Yes
Tags: Array, String, Design, Iterator

Problem Description

Design and implement a data structure for a compressed string iterator. The given compressed string will be in the form of each letter followed by a positive integer representing the number of this letter existing in the original uncompressed string.

Implement the StringIterator class:

* `next()` Returns **the next character** if the original string still has uncompressed characters, otherwise returns a **white space**. * `hasNext()` Returns true if there is any letter needs to be uncompressed in the original string, otherwise returns `false`.

Example 1:

Input ["StringIterator", "next", "next", "next", "next", "next", "next", "hasNext", "next", "hasNext"] [["L1e2t1C1o1d1e1"], [], [], [], [], [], [], [], [], []] **Output** [null, "L", "e", "e", "t", "C", "o", true, "d", true] **Explanation** StringIterator stringIterator = new StringIterator("L1e2t1C1o1d1e1"); stringIterator.next(); // return "L" stringIterator.next(); // return "e" stringIterator.next(); // return "e" stringIterator.next(); // return "t" stringIterator.next(); // return "C" stringIterator.next(); // return "o" stringIterator.hasNext(); // return True stringIterator.next(); // return "d" stringIterator.hasNext(); // return True

Constraints:

* `1 <= compressedString.length <= 1000` * `compressedString` consists of lower-case an upper-case English letters and digits. * The number of a single character repetitions in `compressedString` is in the range `[1, 10^9]` * At most `100` calls will be made to `next` and

`hasNext`.

Code Snippets

C++:

```
class StringIterator {
public:
StringIterator(string compressedString) {

}

char next() {

}

bool hasNext() {

}
};

/**
* Your StringIterator object will be instantiated and called as such:
* StringIterator* obj = new StringIterator(compressedString);
* char param_1 = obj->next();
* bool param_2 = obj->hasNext();
*/
```

Java:

```
class StringIterator {

public StringIterator(String compressedString) {

}

public char next() {

}

public boolean hasNext() {
```

```
}
}

/**
* Your StringIterator object will be instantiated and called as such:
* StringIterator obj = new StringIterator(compressedString);
* char param_1 = obj.next();
* boolean param_2 = obj.hasNext();
*/
```

Python3:

```
class StringIterator:

def __init__(self, compressedString: str):


def next(self) -> str:


def hasNext(self) -> bool:



# Your StringIterator object will be instantiated and called as such:
# obj = StringIterator(compressedString)
# param_1 = obj.next()
# param_2 = obj.hasNext()
```