Problem 1622: Fancy Sequence

Problem Information

Difficulty: Hard
Acceptance Rate: 18.02%
Paid Only: No
Tags: Math, Design, Segment Tree

Problem Description

Write an API that generates fancy sequences using the `append`, `addAll`, and `multAll` operations.

Implement the `Fancy` class:

* `Fancy()` Initializes the object with an empty sequence. * `void append(val)` Appends an integer `val` to the end of the sequence. * `void addAll(inc)` Increments all existing values in the sequence by an integer `inc`. * `void multAll(m)` Multiplies all existing values in the sequence by an integer `m`. * `int getIndex(idx)` Gets the current value at index `idx` (0-indexed) of the sequence **modulo** `109 + 7`. If the index is greater or equal than the length of the sequence, return `-1`.

Example 1:

Input ["Fancy", "append", "addAll", "append", "multAll", "getIndex", "addAll", "append", "multAll", "getIndex", "getIndex", "getIndex"] [[], [2], [3], [7], [2], [0], [3], [10], [2], [0], [1], [2]] **Output** [null, null, null, null, null, 10, null, null, null, 26, 34, 20] **Explanation** Fancy fancy = new Fancy(); fancy.append(2); // fancy sequence: [2] fancy.addAll(3); // fancy sequence: [2+3] -> [5] fancy.append(7); // fancy sequence: [5, 7] fancy.multAll(2); // fancy sequence: [5*2, 7*2] -> [10, 14] fancy.getIndex(0); // return 10 fancy.addAll(3); // fancy sequence: [10+3, 14+3] -> [13, 17] fancy.append(10); // fancy sequence: [13, 17, 10] fancy.multAll(2); // fancy sequence: [13*2, 17*2, 10*2] -> [26, 34, 20] fancy.getIndex(0); // return 26 fancy.getIndex(1); // return 34 fancy.getIndex(2); // return 20

Constraints:

* `1 <= val, inc, m <= 100` * `0 <= idx <= 105` * At most `105` calls total will be made to `append`, `addAll`, `multAll`, and `getIndex`.

Code Snippets

C++:

```
class Fancy {
public:
Fancy() {

}

void append(int val) {

}

void addAll(int inc) {

}

void multAll(int m) {

}

int getIndex(int idx) {

}
};

/**
* Your Fancy object will be instantiated and called as such:
* Fancy* obj = new Fancy();
* obj->append(val);
* obj->addAll(inc);
* obj->multAll(m);
* int param_4 = obj->getIndex(idx);
*/
```

Java:

```
class Fancy {

public Fancy() {

}

public void append(int val) {

}

public void addAll(int inc) {

}

public void multAll(int m) {

}

public int getIndex(int idx) {

}
}

/**
* Your Fancy object will be instantiated and called as such:
* Fancy obj = new Fancy();
* obj.append(val);
* obj.addAll(inc);
* obj.multAll(m);
* int param_4 = obj.getIndex(idx);
*/
```

Python3:

```
class Fancy:

def __init__(self):


def append(self, val: int) -> None:


def addAll(self, inc: int) -> None:
```

```
def multAll(self, m: int) -> None:


def getIndex(self, idx: int) -> int:



# Your Fancy object will be instantiated and called as such:
# obj = Fancy()
# obj.append(val)
# obj.addAll(inc)
# obj.multAll(m)
# param_4 = obj.getIndex(idx)
```