Problem 1381: Design a Stack With Increment Operation

Problem Information

Difficulty: Medium
Acceptance Rate: 80.03%
Paid Only: No
Tags: Array, Stack, Design

Problem Description

Design a stack that supports increment operations on its elements.

Implement the `CustomStack` class:

* `CustomStack(int maxSize)` Initializes the object with `maxSize` which is the maximum number of elements in the stack. * `void push(int x)` Adds `x` to the top of the stack if the stack has not reached the `maxSize`. * `int pop()` Pops and returns the top of the stack or `-1` if the stack is empty. * `void inc(int k, int val)` Increments the bottom `k` elements of the stack by `val`. If there are less than `k` elements in the stack, increment all the elements in the stack.

Example 1:

Input ["CustomStack","push","push","pop","push","push","push","increment","increment","pop","pop","pop","pop"] [[3],[1],[2],[],[2],[3],[4],[5,100],[2,100],[],[],[],[]] **Output** [null,null,null,2,null,null,null,null,null,103,202,201,-1] **Explanation** CustomStack stk = new CustomStack(3); // Stack is Empty [] stk.push(1); // stack becomes [1] stk.push(2); // stack becomes [1, 2] stk.pop(); // return 2 --> Return top of the stack 2, stack becomes [1] stk.push(2); // stack becomes [1, 2] stk.push(3); // stack becomes [1, 2, 3] stk.push(4); // stack still [1, 2, 3], Do not add another elements as size is 4 stk.increment(5, 100); // stack becomes [101, 102, 103] stk.increment(2, 100); // stack becomes [201, 202, 103] stk.pop(); // return 103 --> Return top of the stack 103, stack becomes [201, 202] stk.pop(); // return 202 --> Return top of the stack 202, stack becomes [201] stk.pop(); // return 201 --> Return top of the stack 201, stack becomes [] stk.pop(); // return -1 --> Stack is empty return -1.

Constraints:

* `1 <= maxSize, x, k <= 1000` * `0 <= val <= 100` * At most `1000` calls will be made to each method of `increment`, `push` and `pop` each separately.

Code Snippets

C++:

```
class CustomStack {
public:
CustomStack(int maxSize) {

}

void push(int x) {

}

int pop() {

}

void increment(int k, int val) {

}
};

/**
* Your CustomStack object will be instantiated and called as such:
* CustomStack* obj = new CustomStack(maxSize);
* obj->push(x);
* int param_2 = obj->pop();
* obj->increment(k,val);
*/
```

Java:

```
class CustomStack {

public CustomStack(int maxSize) {
```

```
}

public void push(int x) {

}

public int pop() {

}

public void increment(int k, int val) {

}
}

/**
* Your CustomStack object will be instantiated and called as such:
* CustomStack obj = new CustomStack(maxSize);
* obj.push(x);
* int param_2 = obj.pop();
* obj.increment(k,val);
*/
```

Python3:

```
class CustomStack:

def __init__(self, maxSize: int):


def push(self, x: int) -> None:


def pop(self) -> int:


def increment(self, k: int, val: int) -> None:



# Your CustomStack object will be instantiated and called as such:
```

```
# obj = CustomStack(maxSize)
# obj.push(x)
# param_2 = obj.pop()
# obj.increment(k,val)
```