Implement a simple stack that accepts the following commands and performs the operations associated with them:

- push k: Push integer k onto the top of the stack.
- pop: Pop the top element from the stack.
- inc e k: Add k to each of the bottom e elements of the stack.

Function Description
Complete the function *superStack* in the below code snippet. The function must create an empty stack and perform each of the operations in order. After each operation is performed, print the value of the stack's top element on a new line. If the stack is empty, print EMPTY instead.

superStack has the following parameter(s):
operations[operations[0],...operations[n-1]]: an array of strings

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;

public class Solution {
     /*
     * Complete the function below.
     */
    static void superStack(String[] operations) {

    }

    public static void main(String[] args)  {
        Scanner in = new Scanner(System.in);
        int operations_size = 0;
        operations_size = Integer.parseInt(in.nextLine().trim());

        String[] operations = new String[operations_size];
        for(int i = 0; i < operations_size; i++) {
            String operations_item;
            try {
                operations_item = in.nextLine();
            } catch (Exception e) {
                operations_item = null;
            }
            operations[i] = operations_item;
        }
        superStack(operations);
    }
}
```

Constraints

- 1 ≤ n ≤ 2 × 105
- -109 ≤ k ≤ 109
- 1 ≤ e ≤ |S|, where |S| is the size of the stack at the time of the operation.
- It is guaranteed that pop is never called on an empty stack.

Input Format for Custom Testing

Input from stdin will be processed as follows and passed to the function.

- The first line contains an integer n, the size of the array operations.
- The next n lines each contain an element operations[i] .

Sample Case 0

Sample Input 0

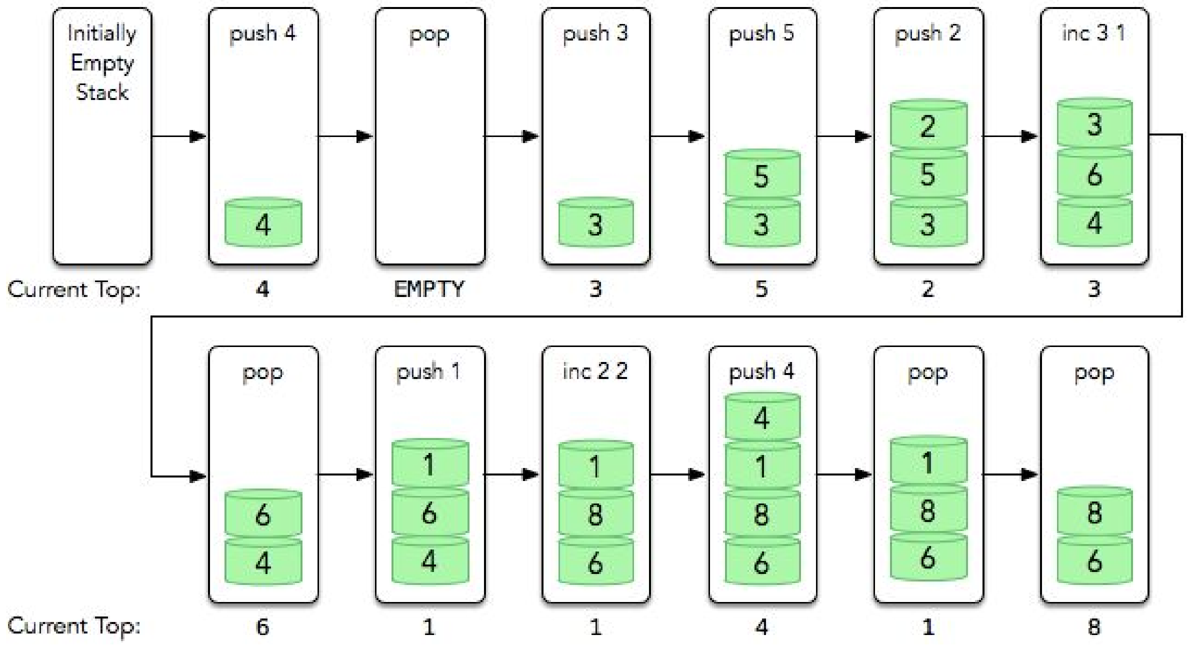
```
12
push 4
pop
push 3
push 5
push 2
inc 3 1
pop
push 1
inc 2 2
push 4
pop
pop
```

Sample Output 0

```
4
EMPTY
3
5
2
3
6
1
```

```
1
4
1
8
```

Explanation 0
The diagram below depicts the stack after each operation:



After each operation, print the value denoted by Current Top on a new line.

In other words, start with an empty stack, S, expressed as an array where the leftmost element is the bottom of the stack and the rightmost element is its top. Perform the following sequence of n = 12 operations as given in the operations array:

1. push 4: Push 4 onto the top of the stack, so S = [4]. Print the top (rightmost) element, 4, on a new line.
2. pop: Pop the top element from the stack, so S = []. Because the stack is now empty, print EMPTY on a new line.
3. push 3: Push 3 onto the top of the stack, S = [3]. Print 3, and the top of the stack after each of the following operations.
4. push 5: Push 5 onto the top of the stack, S = [3, 5].
5. push 2: Push 2 onto the top of the stack, S = [3, 5, 2].
6. inc 3 1: Add k = 1 to bottom e = 3 elements of the stack, S = [4, 6, 3].
7. pop: Pop the top element from the stack, S = [4, 6].
8. push 1: Push 1 onto the top of the stack, S = [4, 6, 1].
9. inc 2 2: Add k = 2 to bottom e = 2 elements of the stack, S = [6, 8, 1].
10. push 4: Push 4 onto the top of the stack, S = [6, 8, 1, 4].
11. pop: Pop the top element from the stack, S = [6, 8, 1].

12. pop: Pop the top element from the stack, S = [6, 8].