

### 59) Build an Array With Stack Operations

You are given an integer array `target` and an integer `n`.

You have an empty stack with the two following operations:

- "Push": pushes an integer to the top of the stack.
- "Pop": removes the integer on the top of the stack.

You also have a stream of the integers in the range `[1, n]`.

Use the two stack operations to make the numbers in the stack (from the bottom to the top) equal to `target`. You should follow the following rules:

- If the stream of the integers is not empty, pick the next integer from the stream and push it to the top of the stack.
- If the stack is not empty, pop the integer at the top of the stack.
- If, at any moment, the elements in the stack (from the bottom to the top) are equal to `target`, do not read new integers from the stream and do not do more operations on the stack.

Return the stack operations needed to build `target` following the mentioned rules. If there are multiple valid answers, return any of them.

Example 1:

Input: `target = [1,3]`, `n = 3`

Output: `["Push","Push","Pop","Push"]`

CODE:

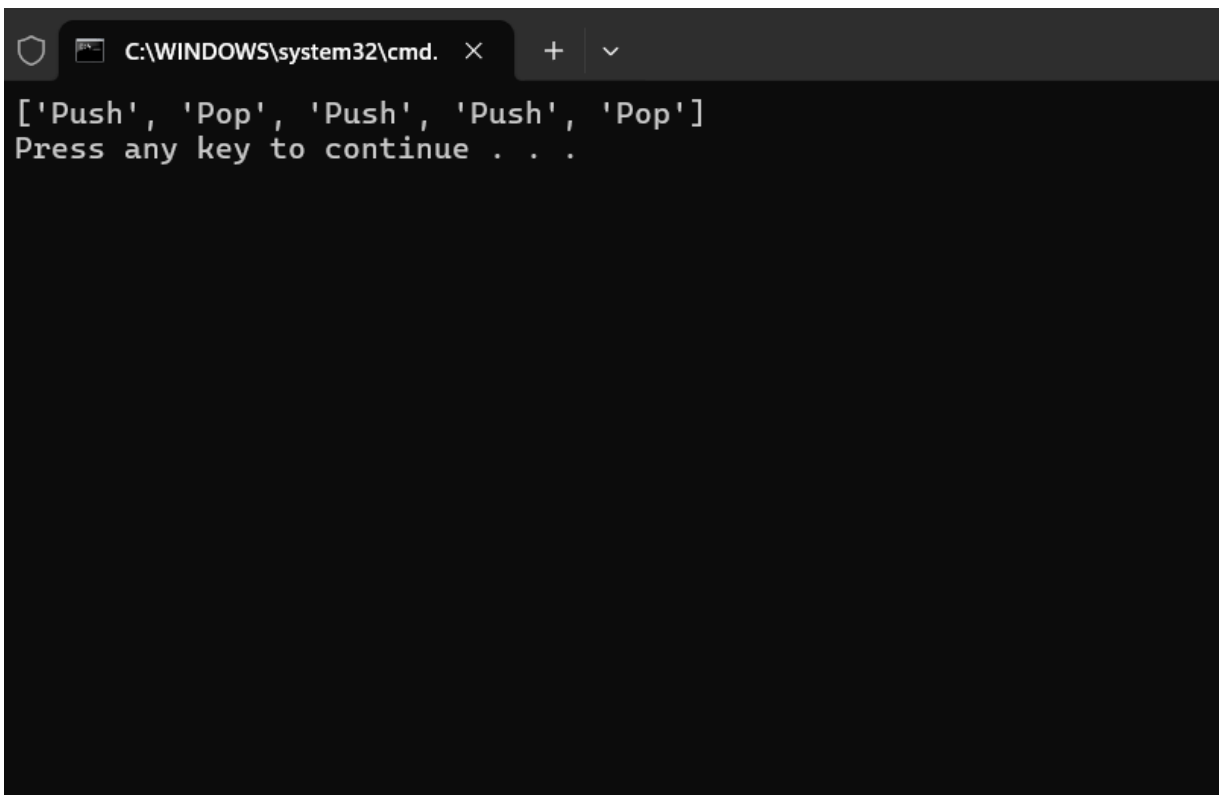
```
def buildArray(target, n):
    stack_ops = []
    stack = []

    for num in range(1, n+1):
        if not stack:
            stack.append(num)
            stack_ops.append("Push")
        else:
            if stack[-1] == target[len(stack)]:
                break
            else:
                stack.append(num)
                stack_ops.append("Push")
        if stack == target:
            break
        elif stack[-1] == target[len(stack)-1]:
            stack.pop()
            stack_ops.append("Pop")

    return stack_ops

target = [1, 3]
n = 3
print(buildArray(target, n))
```

OUTPUT:



A screenshot of a Windows command prompt window. The title bar shows the path 'C:\WINDOWS\system32\cmd.' and standard window controls. The command prompt displays the following text:

```
['Push', 'Pop', 'Push', 'Push', 'Pop']  
Press any key to continue . . .
```

TIME COMPLEXITY :  $O(n)$