Stock Span in C++

```
#include <iostream>
#include <vector>
#include <stack>

using namespace std;

void stockSpan(vector<int>& arr) {
    stack<int> s;
    s.push(0); // Push index of the first element

    for (int i = 0; i < arr.size(); i++) {
        // Pop elements from stack while the current price is greater than the price of the element at the top of the stack
        while (!s.empty() && arr[s.top()] <= arr[i]) {
            s.pop();
        }

        // Calculate span (i - index at top of stack after popping or i + 1 if stack is empty)
        int span = s.empty() ? (i + 1) : (i - s.top());

        // Print the span
        cout << span << " ";

        // Push the current index onto the stack
        s.push(i);
    }
}

int main() {
    // Test case: array of stock prices
    vector<int> arr = {15, 13, 12, 14, 15};
    stockSpan(arr);
    cout << endl;

    return 0;
}
```

For each day i, the span is: The number of consecutive previous days (including the current day) for which the price was **less than or equal to today's price**.

You're maintaining a stack of **indices**, and for each price:

- You **pop** indices from the stack if the current price is higher than the price at the stack's top.
- The **span** is then i - s.top() or i + 1 if the stack is empty.

🧪 Input:
arr = {15, 13, 12, 14, 15}

📋 **Dry Run Table:**

Day (i)	Price	Stack (indices)	Stack (values)	Span	Explanation
0	15	[ ]	[ ]	1	Stack empty → span = 0 + 1
		[0]	[15]		Push index 0
1	13	[0]	[15]	1	13 < 15 → span = 1 - 0
		[0, 1]	[15, 13]		
2	12	[0, 1]	[15, 13]	1	12 < 13 → span = 2 - 1
		[0, 1, 2]	[15, 13, 12]		
3	14	[0, 1, 2] → pop 2, 1	[15]	3	14 > 13 & 12 → span = 3 - 0
		[0, 3]	[15, 14]		
4	15	[0, 3] → pop 3, 0	[ ]	5	15 >= 14, 15 → stack empty → span = 4 + 1
		[4]	[15]		

✅ **Output: 1 1 1 3 5**

1 1 1 3 5