Climbing Stairs in C++

```
#include <iostream>
#include <vector>
#include <climits> // For INT_MAX

using namespace std;

void printMinSteps(vector<int>& arr) {
    int n = arr.size();
    vector<int> dp(n + 1, INT_MAX); // Use INT_MAX for
initialization

    dp[n] = 0; // Base case: 0 steps needed from the end

    for (int i = n - 1; i >= 0; i--) {
        if (arr[i] > 0) {
            int minSteps = INT_MAX;
            for (int j = 1; j <= arr[i] && (i + j) < dp.size(); j++)
{
                if (dp[i + j] != INT_MAX) {
                    minSteps = min(minSteps, dp[i + j]);
                }
            }
            if (minSteps != INT_MAX) {
                dp[i] = minSteps + 1;
            }
        }
    }

    // Printing the dp array
    for (int i = 0; i < dp.size(); i++) {
        cout << " " << dp[i];
    }
    cout << endl;
}

int main() {
    vector<int> arr = {1, 5, 2, 3, 1};
    printMinSteps(arr);

    return 0;
}
```

Input:

- arr = {1, 5, 2, 3, 1}

Initialization:

- n = 5 (size of arr)
- dp = {INT_MAX, INT_MAX, INT_MAX, INT_MAX, INT_MAX, 0} (base case: dp[n] = 0)

Iterations:

Step 1: Start from i = 4:

- arr[4] = 1 → Maximum jump = 1
- Valid jump: j = 1
    - dp[4] = min(dp[5]) + 1 = 0 + 1 = 1
- Updated dp: {INT_MAX, INT_MAX, INT_MAX, INT_MAX, 1, 0}

Step 2: i = 3:

- arr[3] = 3 → Maximum jump = 3
- Valid jumps: j = 1, 2
    - dp[3] = min(dp[4], dp[5]) + 1 = min(1, 0) + 1 = 1
- Updated dp: {INT_MAX, INT_MAX, INT_MAX, 1, 1, 0}

Step 3: i = 2:

- arr[2] = 2 → Maximum jump = 2
- Valid jumps: j = 1, 2
    - dp[2] = min(dp[3], dp[4]) + 1 = min(1, 1) + 1 = 2
- Updated dp: {INT_MAX, INT_MAX, 2, 1, 1, 0}

Step 4: i = 1:

- arr[1] = 5 → Maximum jump = 5
- Valid jumps: j = 1, 2, 3, 4
    - dp[1] = min(dp[2], dp[3], dp[4], dp[5]) + 1 = min(2, 1, 1, 0) + 1 = 1
- Updated dp: {INT_MAX, 1, 2, 1, 1, 0}

	Step 5: i = 0: • arr[0] = 1 → Maximum jump = 1 • Valid jump: j = 1 ○ dp[0] = min(dp[1]) + 1 = 1 + 1 = 2 • Updated dp: {2, 1, 2, 1, 1, 0}
Output:- 🕐 Printed dp: 2 1 2 1 1 0 🕐 The minimum steps to reach the end starting from index 0 is dp[0] = 2.	