Count Distinct Subsequence C++

```
#include <iostream>
using namespace std;

int countValleysAndMountains(int n) {
    int dp[n + 1] = {0}; // Initialize the array with zeros
    dp[0] = 1; // Base case: empty sequence
    dp[1] = 1; // Sequence of length 1: either V or M

    for (int i = 2; i <= n; i++) {
        int valleys = 0;
        int mountains = i - 1;

        while (mountains >= 0) {
            dp[i] += dp[valleys] * dp[mountains];
            valleys++;
            mountains--;
        }
    }

    return dp[n];
}

int main() {
    int n = 5;
    cout << countValleysAndMountains(n) << endl;
    return 0;
}
```

Dry Run Example for n = 5

Let's break down the example when n = 5.

1. **Initialization**:
    - dp[0] = 1 (One way to form an empty sequence).
    - dp[1] = 1 (One way to form a sequence of length 1, either "V" or "M").
2. **Filling dp[2] to dp[5]**:
    - **For i = 2**:
        - dp[2] = dp[0] * dp[1] + dp[1] * dp[0]
        - dp[2] = 1 * 1 + 1 * 1 = 2
    - **For i = 3**:
        - dp[3] = dp[0] * dp[2] + dp[1] * dp[1] + dp[2] * dp[0]
        - dp[3] = 1 * 2 + 1 * 1 + 2 * 1 = 5
    - **For i = 4**:
        - dp[4] = dp[0] * dp[3] + dp[1] * dp[2] + dp[2] * dp[1] + dp[3] * dp[0]
        - dp[4] = 1 * 5 + 1 * 2 + 2 * 1 + 5 * 1 = 14
    - **For i = 5**:
        - dp[5] = dp[0] * dp[4] + dp[1] * dp[3] + dp[2] * dp[2] + dp[3] * dp[1] + dp[4] * dp[0]
        - dp[5] = 1 * 14 + 1 * 5 + 2 * 2 + 5 * 1 + 14 * 1 = 42
3. **Output**:
    - The final value of dp[5] is 42, which is the number of valid valley-mountain sequences of length 5.

Output:-

```
42
```