Coin Change Combination in C++

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

using namespace std;

int main() {
    vector<int> arr = {2, 3, 5};
    int amt = 7;
    vector<int> dp(amt + 1, 0);
    dp[0] = 1; // Base case: 1 way to make amount 0 (using no coins)

    for (int i = 0; i < arr.size(); i++) {
        for (int j = arr[i]; j <= amt; j++) {
            dp[j] += dp[j - arr[i]];
        }
    }

    cout << dp[amt] << endl; // Output the number of combinations for amount `amt`

    return 0;
}
```

Input:

- arr = {2, 3, 5}
- amt = 7

Initialization:

- dp = {1, 0, 0, 0, 0, 0, 0, 0} (size = amt + 1, initialized to 0 except dp[0] = 1).

Iterations:

Step 1: Using Coin 2 (arr[0]):

- For each j from 2 to 7:
    - dp[j] += dp[j - 2]
- Updates:

  dp[2] = dp[2] + dp[0] = 0 + 1 = 1
  dp[3] = dp[3] + dp[1] = 0 + 0 = 0
  dp[4] = dp[4] + dp[2] = 0 + 1 = 1
  dp[5] = dp[5] + dp[3] = 0 + 0 = 0
  dp[6] = dp[6] + dp[4] = 0 + 1 = 1
  dp[7] = dp[7] + dp[5] = 0 + 0 = 0

- dp = {1, 0, 1, 0, 1, 0, 1, 0}

Step 2: Using Coin 3 (arr[1]):

- For each j from 3 to 7:
    - dp[j] += dp[j - 3]
- Updates:

  dp[3] = dp[3] + dp[0] = 0 + 1 = 1
  dp[4] = dp[4] + dp[1] = 1 + 0 = 1
  dp[5] = dp[5] + dp[2] = 0 + 1 = 1
  dp[6] = dp[6] + dp[3] = 1 + 1 = 2
  dp[7] = dp[7] + dp[4] = 0 + 1 = 1

- dp = {1, 0, 1, 1, 1, 1, 2, 1}

Step 3: Using Coin 5 (arr[2]):

- For each j from 5 to 7:
    - dp[j] += dp[j - 5]
- Updates:

  dp[5] = dp[5] + dp[0] = 1 + 1 = 2
  dp[6] = dp[6] + dp[1] = 2 + 0 = 2
  dp[7] = dp[7] + dp[2] = 1 + 1 = 2

	• dp = {1, 0, 1, 1, 1, 2, 2, 2} **Final DP Array:** dp = {1, 0, 1, 1, 1, 2, 2, 2} **Output:** • dp[amt] = dp[7] = 2 • There are **2 ways** to form amount 7 using coins {2, 3, 5}.
Output:- 2	