Burst Balloons In C++

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

int sol(int arr[], int n) {
    int dp[n][n];

    // Initialize the dp array with zeros
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            dp[i][j] = 0;
        }
    }

    for (int g = 0; g < n; g++) {
        for (int i = 0, j = g; j < n; i++, j++) {
            int maxCoins = INT_MIN;
            for (int k = i; k <= j; k++) {
                int left = (k == i) ? 0 : dp[i][k - 1];
                int right = (k == j) ? 0 : dp[k + 1]
[j];
                int val = (i == 0 ? 1 : arr[i - 1]) *
arr[k] * (j == n - 1 ? 1 : arr[j + 1]);
                int total = left + right + val;
                maxCoins = max(maxCoins,
total);
            }
            dp[i][j] = maxCoins;
        }
    }
    return dp[0][n - 1];
}

int main() {
    int arr[] = {2, 3, 5};
    int n = sizeof(arr) / sizeof(arr[0]);
    cout << sol(arr, n) << endl;
    return 0;
}
```

Step-by-Step Dry Run

Given input:

int arr[] = {2, 3, 5};

Here, the balloons' values are 2, 3, and 5.

- We initialize the dp array with zeros.

First Iteration (gap = 0, considering only single balloons):

- **i = 0, j = 0**:
  - maxCoins = INT_MIN
  - Only one balloon at index 0, so the value for bursting it is 1 * 2 * 1 = 2. The result is stored in dp[0][0].
- **i = 1, j = 1**:
  - maxCoins = INT_MIN
  - Only one balloon at index 1, so the value for bursting it is 1 * 3 * 1 = 3. The result is stored in dp[1][1].
- **i = 2, j = 2**:
  - maxCoins = INT_MIN
  - Only one balloon at index 2, so the value for bursting it is 1 * 5 * 1 = 5. The result is stored in dp[2][2].

Second Iteration (gap = 1, considering two consecutive balloons):

- **i = 0, j = 1**:
  - We check two possible balloons to burst, k = 0 and k = 1.
  - If we burst k = 0 first, the coins obtained are:
    - Left: 0, Right: dp[1][1] = 3, Value from bursting: 1 * 2 * 3 = 6, so total = 6 + 3 = 9.
  - If we burst k = 1 first, the coins obtained are:
    - Left: dp[0][0] = 2, Right: 0, Value from bursting: 1 * 3 * 5 = 15, so total = 2 + 15 = 17.
  - We store the maximum value 17 in dp[0][1].
- **i = 1, j = 2**:
  - We check two possible balloons to burst, k = 1 and k = 2.

	○ If we burst k = 1 first, the coins obtained are: ▪ Left: 0, Right: dp[2][2] = 5, Value from bursting: 2 * 3 * 5 = 30, so total = 30 + 5 = 35. ○ If we burst k = 2 first, the coins obtained are: ▪ Left: dp[1][1] = 3, Right: 0, Value from bursting: 2 * 3 * 5 = 30, so total = 3 + 30 = 33. ○ We store the maximum value 35 in dp[1][2]. **Third Iteration (gap = 2, considering the whole array):** • **i = 0, j = 2**: ○ We check three possible balloons to burst, k = 0, k = 1, and k = 2. ○ If we burst k = 0 first: ▪ Left: 0, Right: dp[1][2] = 35, Value from bursting: 1 * 2 * 5 = 10, so total = 10 + 35 = 45. ○ If we burst k = 1 first: ▪ Left: dp[0][0] = 2, Right: dp[2][2] = 5, Value from bursting: 1 * 3 * 5 = 15, so total = 2 + 15 + 5 = 22. ○ If we burst k = 2 first: ▪ Left: dp[0][1] = 17, Right: 0, Value from bursting: 1 * 3 * 5 = 15, so total = 17 + 15 = 32. ○ The maximum value 45 is stored in dp[0][2]. **Final Result:** • The value in dp[0][2] (maximum coins from bursting all balloons) is 45.
Output:- 45	