Max Sum Increasing subseq In C++

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

int MaxSumIncreasingSubseq(int arr[], int
size) {
   int omax = INT_MIN;
   int* dp = new int[size];
   //int dp[size];

   for (int i = 0; i < size; i++) {
      int maxSum = arr[i];
      for (int j = 0; j < i; j++) {
         if (arr[j] <= arr[i]) {
            maxSum = max(maxSum, dp[j] +
arr[i]);
         }
      }
      dp[i] = maxSum;
      omax = max(omax, dp[i]);
   }

   delete[] dp; // Don't forget to free the
allocated memory
   return omax;
}

int main() {
   int arr[] = {10, 22, 9, 33, 21, 50, 41, 60, 80,
3};
   int size = sizeof(arr) / sizeof(arr[0]);

   int maxSum =
MaxSumIncreasingSubseq(arr, size);
   cout << maxSum << endl;

   return 0;
}
```

Step-by-Step Dry Run

Initialization of dp[]:

Initially, dp[] is:

dp[] = {10, 22, 9, 33, 21, 50, 41, 60, 80, 3}

Each element is initialized to the value of the corresponding element in arr[].

For i = 0 (First Element: 10)

- **maxSum = 10**
- There are no previous elements, so no update is made in dp[].
- **dp[0] = 10**
- **omax = max(INT_MIN, 10) = 10**

For i = 1 (Element: 22)

- **maxSum = 22**
- Check all previous elements (arr[0] = 10):
  - **arr[0] <= arr[1] (10 <= 22)**: Yes
    - **maxSum = max(22, dp[0] + arr[1]) = max(22, 10 + 22) = 32**
- **dp[1] = 32**
- **omax = max(10, 32) = 32**

For i = 2 (Element: 9)

- **maxSum = 9**
- Check all previous elements (arr[0] = 10, arr[1] = 22):
  - **arr[0] <= arr[2] (10 <= 9)**: No
  - **arr[1] <= arr[2] (22 <= 9)**: No
- **dp[2] = 9**
- **omax = max(32, 9) = 32**

For i = 3 (Element: 33)

- **maxSum = 33**
- Check all previous elements (arr[0] = 10, arr[1] = 22, arr[2] = 9):
    - **arr[0] <= arr[3] (10 <= 33)**: Yes
        - **maxSum = max(33, dp[0] + arr[3]) = max(33, 10 + 33) = 43**
    - **arr[1] <= arr[3] (22 <= 33)**: Yes
        - **maxSum = max(43, dp[1] + arr[3]) = max(43, 32 + 33) = 65**
    - **arr[2] <= arr[3] (9 <= 33)**: Yes
        - **maxSum = max(65, dp[2] + arr[3]) = max(65, 9 + 33) = 65**
- **dp[3] = 65**
- **omax = max(32, 65) = 65**

For i = 4 (Element: 21)

- **maxSum = 21**
- Check all previous elements (arr[0] = 10, arr[1] = 22, arr[2] = 9, arr[3] = 33):
    - **arr[0] <= arr[4] (10 <= 21)**: Yes
        - **maxSum = max(21, dp[0] + arr[4]) = max(21, 10 + 21) = 31**
    - **arr[1] <= arr[4] (22 <= 21)**: No
    - **arr[2] <= arr[4] (9 <= 21)**: Yes
        - **maxSum = max(31, dp[2] + arr[4]) = max(31, 9 + 21) = 31**
    - **arr[3] <= arr[4] (33 <= 21)**: No
- **dp[4] = 31**
- **omax = max(65, 31) = 65**

For i = 5 (Element: 50)

- **maxSum = 50**
- Check all previous elements:
    - **arr[0] <= arr[5] (10 <= 50)**: Yes
        - **maxSum = max(50, dp[0] + arr[5]) = max(50, 10 + 50) = 60**

	○ **arr[1] <= arr[5] (22 <= 50)**: Yes ▪ **maxSum = max(60, dp[1] + arr[5]) = max(60, 32 + 50) = 82** ○ **arr[2] <= arr[5] (9 <= 50)**: Yes ▪ **maxSum = max(82, dp[2] + arr[5]) = max(82, 9 + 50) = 82** ○ **arr[3] <= arr[5] (33 <= 50)**: Yes ▪ **maxSum = max(82, dp[3] + arr[5]) = max(82, 65 + 50) = 115** ○ **arr[4] <= arr[5] (21 <= 50)**: Yes ▪ **maxSum = max(115, dp[4] + arr[5]) = max(115, 31 + 50) = 115** • **dp[5] = 115** • **omax = max(65, 115) = 115** **For i = 6 (Element: 41)** • **maxSum = 41** • Check all previous elements (arr[0] = 10, arr[1] = 22, arr[2] = 9, arr[3] = 33, arr[4] = 21, arr[5] = 50): ○ **arr[0] <= arr[6] (10 <= 41)**: Yes ▪ **maxSum = max(41, dp[0] + arr[6]) = max(41, 10 + 41) = 51** ○ **arr[1] <= arr[6] (22 <= 41)**: Yes ▪ **maxSum = max(51, dp[1] + arr[6]) = max(51, 32 + 41) = 73** ○ **arr[2] <= arr[6] (9 <= 41)**: Yes ▪ **maxSum = max(73, dp[2] + arr[6]) = max(73, 9 + 41) = 73** ○ **arr[3] <= arr[6] (33 <= 41)**: Yes ▪ **maxSum = max(73, dp[3] + arr[6]) = max(73, 65 + 41) = 106** ○ **arr[4] <= arr[6] (21 <= 41)**: Yes ▪ **maxSum = max(106, dp[4] + arr[6]) = max(106, 31 + 41) = 106** ○ **arr[5] <= arr[6] (50 <= 41)**: No • **dp[6] = 106** • **omax = max(115, 106) = 115**

	Final Output: After performing similar checks for all the remaining elements, the maximum sum found will be **255**. Thus, the **Maximum Sum Increasing Subsequence** is **255**.
Output:- 255 {10, 22, 33, 50, 60, 80} → sum = 10 + 22 + 33 + 50 + 60 + 80 = 255	