|  |  |
| --- | --- |
| **Subset Sum in C++** | |
| #include <iostream>  using namespace std;  // Function to calculate subset sums recursively  void subsetSums(int arr[], int l, int r, int sum) {      // Base case: if l exceeds r, print the current sum      if (l > r) {          cout << sum << " ";          return;      }        // Recursive case: include current element arr[l] in the subset sum      subsetSums(arr, l + 1, r, sum + arr[l]);  }  int main() {      // Initialize the array and its length      int arr[] = {5, 4, 3, 5, 4};      int n = sizeof(arr) / sizeof(arr[0]);        // Call the function to calculate subset sums, starting with l=0, r=n-1, and initial sum=0      subsetSums(arr, 0, n - 1, 0);        return 0;  } | Input: int arr[] = {5, 4, 3, 5, 4};  This adds:  5 + 4 + 3 + 5 + 4 = 21  And when l > r, it prints sum, which is 21. 📋 Dry Run Table (for your input):  | **Step** | **l** | **r** | **sum** | **Action** | | --- | --- | --- | --- | --- | | 1 | 0 | 4 | 0 | sum = 0 + arr[0] = 5 | | 2 | 1 | 4 | 5 | sum = 5 + arr[1] = 9 | | 3 | 2 | 4 | 9 | sum = 9 + arr[2] = 12 | | 4 | 3 | 4 | 12 | sum = 12 + arr[3] = 17 | | 5 | 4 | 4 | 17 | sum = 17 + arr[4] = 21 | | 6 | 5 | 4 | 21 | l > r, print 21 and return |  ✅ Final Output: 21 |
| Output:- 21 | |