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;
}
```

Dry Run of subsetSums(arr, 0, 4, 0)

Let's dry run this code using the input array {5, 4, 3, 5, 4}.

Initial Call: subsetSums(arr, 0, 4, 0)

Call 1: subsetSums(arr, 0, 4, 0)

- We include arr[0] which is 5.
  - Next call: subsetSums(arr, 1, 4, 5)

Call 2: subsetSums(arr, 1, 4, 5)

- We include arr[1] which is 4.
  - Next call: subsetSums(arr, 2, 4, 9)

Call 3: subsetSums(arr, 2, 4, 9)

- We include arr[2] which is 3.
  - Next call: subsetSums(arr, 3, 4, 12)

Call 4: subsetSums(arr, 3, 4, 12)

- We include arr[3] which is 5.
  - Next call: subsetSums(arr, 4, 4, 17)

Call 5: subsetSums(arr, 4, 4, 17)

- We include arr[4] which is 4.
  - Next call: subsetSums(arr, 5, 4, 21) — **Base case reached**, prints 21.

Backtracking and Generating Other Subsets

Now, the recursion starts backtracking. The function will explore subsets where elements are **not** included.

Call 6: subsetSums(arr, 4, 4, 17) (skip arr[4])

- We **skip** arr[4] (i.e., do not add it to the

subset).
    - Next call: subsetSums(arr, 5, 4, 17) — **Base case reached**, prints 17.

Call 7: subsetSums(arr, 3, 4, 12) (skip arr[3])

- We **skip** arr[3] (i.e., do not add it to the subset).
    - Next call: subsetSums(arr, 4, 4, 12)

Call 8: subsetSums(arr, 4, 4, 12) (skip arr[4])

- We **skip** arr[4].
    - Next call: subsetSums(arr, 5, 4, 12) — **Base case reached**, prints 12.

Call 9: subsetSums(arr, 2, 4, 9) (skip arr[2])

- We **skip** arr[2] (i.e., do not add it to the subset).
    - Next call: subsetSums(arr, 3, 4, 9)

Call 10: subsetSums(arr, 3, 4, 9) (skip arr[3])

- We **skip** arr[3] (i.e., do not add it to the subset).
    - Next call: subsetSums(arr, 4, 4, 9)

Call 11: subsetSums(arr, 4, 4, 9) (skip arr[4])

- We **skip** arr[4].
    - Next call: subsetSums(arr, 5, 4, 9) — **Base case reached**, prints 9.

Call 12: subsetSums(arr, 1, 4, 5) (skip arr[1])

- We **skip** arr[1] (i.e., do not add it to the subset).
    - Next call: subsetSums(arr, 2, 4, 5)

Call 13: subsetSums(arr, 2, 4, 5) (skip arr[2])

	• We **skip** arr[2].
	  ○ Next call: subsetSums(arr, 3, 4, 5)
	
	Call 14: subsetSums(arr, 3, 4, 5) (skip arr[3])
	
	• We **skip** arr[3].
	  ○ Next call: subsetSums(arr, 4, 4, 5)
	
	Call 15: subsetSums(arr, 4, 4, 5) (skip arr[4])
	
	• We **skip** arr[4].
	  ○ Next call: subsetSums(arr, 5, 4, 5) — **Base case reached**, prints 5.
	
	Final Output
	
	The program will print the subset sums of the array {5, 4, 3, 5, 4}:
	
	21 17 12 9 5

Output:- 21	