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

// global stuff
int n = 5;
// I waste spot 0 to be consistent with the code developed in class/textbook
int weights[] = {-1, 5, 6, 10, 11, 16};
bool includes[] = {false, false, false, false, false, false};
int W = 21;

void printSolutionSubset() {
    cout << "Sum of " << W << ": {";
    for (int i = 1; i <= n; i++) {
        if (includes[i]) {
            cout << weights[i] << " ";
        } // end if
    } // end for
    cout << "}" << endl << endl;
} // end printSolutionSubset

bool promising(int level, int subsetSum, int totalOfRemainingItems) {
    if (subsetSum == W)
        return true;

    if (subsetSum > W || (subsetSum + totalOfRemainingItems < W) || (subsetSum + weights[level] > W)) {
        return false;
    } else {
        return true;
    } // end if

} // end promising

void sumOfSubsets(int level, int subsetSum, int totalOfRemainingItems) {
    // cout << "level="<<level<< " subsetSum="<<subsetSum<<" totalRemaining="<< totalOfRemainingItems<<
    endl;
    if (level >= n)
        return;

    // replaced the for-loop by unrolling it (i.e., performing body twice)
    // consider right child where we include the next weight in the subset
    if (promising(level+1, subsetSum + weights[level+1], totalOfRemainingItems-weights[level+1]) ) {
        includes[level + 1] = true;
        if (subsetSum + weights[level+1] == W) { // solution found
            printSolutionSubset();
        } else {
            sumOfSubsets(level+1, subsetSum + weights[level+1], totalOfRemainingItems-weights[level+1]);
        } // end if
    } // end if promising

    // consider left child where we do NOT include the next weight in the subset
    includes[level+1] = false;
    if (promising(level+1, subsetSum, totalOfRemainingItems-weights[level+1]) ) {
        sumOfSubsets(level+1, subsetSum, totalOfRemainingItems-weights[level+1]);

    } // end if promising

} // end sumOfSubsets


int main() {

    sumOfSubsets(0, 0, 48);
    return 0;
}
```