

# cs330su21-a.sg

[Dashboard](#) / [My courses](#) / [cs330su21-a.sg](#) / [Week 8 \(28 June - 4 July\)](#) / [Programming Assignment 1 - Backtracking](#)

-  [Description](#)


 [Submission](#)

 [Edit](#)

 Submission view

## Grade

Reviewed on Friday, 2 July 2021, 2:49 PM by Automatic grade  
**grade:** 100.00 / 100.00

**Assessment report**  [-]  
[\[+\]](#) **Summary of tests**

Submitted on Friday, 2 July 2021, 2:49 PM ([Download](#))  
q.cpp

```

1  /*!*****
2  \file    q.cpp
3  \author  Goh Wei Zhe
4  \par    DP email: weizhe.goh@digipen.edu
5  \par    Course: CS330
6  \par    Section: A
7  \par    Programming Assignment #1
8  \date   2-07-2021
9
10 \brief Implementation of backtracking algorithm
11
12 *****/
13
14 #include <iostream>
15 #include <vector>
16 #include <numeric> //accumulate
17
18 namespace CS330
19 {
20     namespace subsetsum
21     {
22         bool subset_rec(std::vector<int> const& set, int sum,
23             std::vector<int>& subset, unsigned long index)
24         {
25             int curr_sum = std::accumulate(subset.begin(), subset.end(), 0);
26
27             /*----- Success: find a solution -----*/
28             if (curr_sum == sum)
29             {
30                 for (unsigned long i = 0; i < subset.size(); ++i)
31                 {
32                     std::cout << subset[i] << " ";
33                 }
34
35                 std::cout << " sum " << curr_sum << std::endl;
36                 return true;
37             }
38
39             /*----- Hit the bottom: start backtracking -----*/
40
41             // Add code here
42             // Your code should print out the candidate solution before
43             // return false
44             // The output should be in format (# represents a number):
45             // # # # # sum #
46
47             if (index == set.size())
48             {
49                 if (!subset.empty())
50                 {
51                     for (unsigned long i = 0; i < subset.size(); ++i)
52                     {
53                         std::cout << subset[i] << " ";
54                     }
55
56                     std::cout << " sum " << curr_sum << std::endl;
57                     return false;
58                 }
59             }
60
61             /*----- Backtracking -----*/
62
63             // ADD code here.
64             // You backtracking should stop as soon as the first solution is found
65             //Backtrack
66
67             // left: add nothing {} to the solution
68             subset.push_back(0);
69
70             if (subset_rec(set, sum, subset, index + 1))
71                 return true;
72             else
73                 subset.pop_back();
74
75             // right: add the element set[index] to the solution
76             subset.push_back(set[index]);
77
78             if (subset_rec(set, sum, subset, index + 1))
79                 return true;
80             else
81                 subset.pop_back();
82
83             return false; // continue backtracking
84         }
85
86         std::vector<int> subset_sum(std::vector<int> const& set, int sum)
87         {
88             std::vector<int> subset;
89             CS330::subsetsum::subset_rec(set, sum, subset, 0);
90             return subset;
91         }
92     }
93 }

```

[VPL](#)

◀ Attendance cs330su21-a.sg  
Thursday 1/07/2021 11:00am-  
12:40pm

Jump to...



[cs330su21-a.sg](#)  
[Data retention summary.](#)  
[Get the mobile app](#)