Solution 1

```cpp
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

#include <vector>
using namespace std;

vector<vector<int>> getKnapsackItems(vector<vector<int>> knapsackValues,
                                     vector<vector<int>> items, int weight);

// O(nc) time | O(nc) space
vector<vector<int>> knapsackProblem(vector<vector<int>> items, int capacity) {
  vector<vector<int>> knapsackValues(items.size() + 1,
                                     vector<int>(capacity + 1, 0));
  for (int i = 1; i < items.size() + 1; i++) {
    int currentWeight = items[i - 1][1];
    int currentValue = items[i - 1][0];
    for (int c = 0; c < capacity + 1; c++) {
      if (currentWeight > c) {
        knapsackValues[i][c] = knapsackValues[i - 1][c];
      } else {
        knapsackValues[i][c] =
            max(knapsackValues[i - 1][c],
                knapsackValues[i - 1][c - currentWeight] + currentValue);
      }
    }
  }
  return getKnapsackItems(knapsackValues, items,
                          knapsackValues[items.size()][capacity]);
}

vector<vector<int>> getKnapsackItems(vector<vector<int>> knapsackValues,
                                     vector<vector<int>> items, int weight) {
  vector<vector<int>> solution = {{}, {}};
  int i = knapsackValues.size() - 1;
  int c = knapsackValues[0].size() - 1;
  while (i > 0) {
    if (knapsackValues[i][c] == knapsackValues[i - 1][c]) {
      i--;
    } else {
      solution[1].insert(solution[1].begin(), i - 1);
      c -= items[i - 1][1];
      i--;
    }
    if (c == 0) {
      break;
    }
  }
  solution[0].push_back(weight);
  return solution;
}

```