Solution 1

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

// O(nc) time | O(nc) space
function knapsackProblem(items, capacity) {
  const knapsackValues = [];
  for (let i = 0; i < items.length + 1; i++) {
    const row = new Array(capacity + 1).fill(0);
    knapsackValues.push(row);
  }
  for (let i = 1; i < items.length + 1; i++) {
    const currentWeight = items[i - 1][1];
    const currentValue = items[i - 1][0];
    for (let c = 0; c < capacity + 1; c++) {
      if (currentWeight > c) {
        knapsackValues[i][c] = knapsackValues[i - 1][c];
      } else {
        knapsackValues[i][c] = Math.max(
          knapsackValues[i - 1][c],
          knapsackValues[i - 1][c - currentWeight] + currentValue,
        );
      }
    }
  }
  return [knapsackValues[items.length][capacity], getKnapsackItems(knapsackValues, items)];
}

function getKnapsackItems(knapsackValues, items) {
  const sequence = [];
  let i = knapsackValues.length - 1;
  let c = knapsackValues[0].length - 1;
  while (i > 0) {
    if (knapsackValues[i][c] === knapsackValues[i - 1][c]) {
      i -= 1;
    } else {
      sequence.unshift(i - 1);
      c -= items[i - 1][1];
      i -= 1;
    }
    if (c === 0) break;
  }
  return sequence;
}

exports.knapsackProblem = knapsackProblem;

```