Question 3

Documentation: I briefly discussed this problem with Joseph So.

Every item weighs at least 1 and at most W.

You can only take each item once.

```
globals:
w = array of weights
v = array of valus
W = maxWeight
n = number of items

function solve() {
  _, nodes = solve(0, 0)
  return nodes
}

lookupTable = {}

function solve(weightSoFar, i) {
  if (i >= n) {
    return {0, []}
  }
  key = weightSoFar+":"+i
  cached = lookupTable[key]
  if (cached) {
    return cached
  }

  rightScore, rightNodes = solve(weightSoFar, i+1);

  if (weightSoFar + w[i] <= W) {
    leftScore, nodes = solve(weightSoFar + w[i], i+1);
    if (leftScore > rightScore) {
      return {leftScore+v[i], nodes+[i]};
    }
  }
  val = {rightScore, rightNodes}
  lookupTable[key] = val
  return val
}
```

This runs in $O(nW)$ since it builds up a table of (weight, index) values. Since there's W possible weights and n possible indices `solve` will possibly run nW times. The computations in `solve` take $O(1)$, thus in total $O(nW)$.

Interesting Optional Exercise

You could just take the items and sort them by their value density, and then return the items with the highest value density with the last one partially taken.