In-class exercise

Collaboration level 0 (no restrictions). Open notes.

0-1 Knapsack: Consider the knapsack problem discussed in class: We have a backpack of capacity W and a set of n items, each item with weight $w[i]$ and value $v[i]$. We want to compute the maximal value for packing the backpack with the items. Below is the top-down recursive DP algorithm which fills in the table $table[0..n][0..W]$.

```
OPTKNAPSACKDP($i, x, table$)
    // Assume global variables $v[]$ and $w[]$.
    // RETURNs the max value to pack a knapsack of capacity $x$ using items 1 through $i$.
    if ($x == 0$): return 0
    if ($i \leq 0$): return 0
    IF ($table[i][x] \neq -1$): RETURN $table[i][x]$
    IF $w[i] \leq x$: $with = v[i] +$ OPTKNAPSACKDP($i-1, x-w[i], table$)
    ELSE: with = 0
    $without =$ OPTKNAPSACKDP($i-1, x, table$)
    $table[i][x] = \max\{ with, without \}$
    RETURN $table[i][x]$
```

Assume we have a backpack of capacity $W = 3$, and three items ($n = 3$): a hat of weight 1 and value 4, a ball of weight 2 and value 3, and a bottle of water of weight 3 and value 6.

(a) Draw the tree of recursive calls triggered by $optknapsackDP(3,3)$ (an edge from a to b if a generates a recursive call to b); show this tree on the table below.

	x=0	x=1	x=2	x=3
i=0	0	0	0	0
i=1	0			
i=2	0			
i=3	0			

	x=0	x=1	x=2	x=3
i=0	0	0	0	0
i=1	0			
i=2	0			
i=3	0			

(b) Follow the recursion and calculate the values that will be stored in the table. Only show the values that are actually filled in. Which entries are not filled in?

(c) Number the entries in the table in the order in which they are filled in by the recursion.

(d) Assemble the full solution for $optknapsackDP(3,3)$, and list the entries you visit.