Due at 11:59PM, Monday April 14th on Canvas. Submissions must be ***typed*** and in PDF format. Show your work for full credit!

Each question is about designing a dynamic programming algorithm, based on your backtracking algorithm from the previous homework. First, ***state the recurrence or backtracking algorithm*** from the previous homework. Then, use it to write down a ***dynamic programming algorithm*** in pseudocode. Explain your choices of ***data structure*** (for storing the subproblems) and the ***fill order*** you're using. Each algorithm must be purely ***iterative***, with no recursive calls. Afterwards, analyze the ***runtime*** of your algorithm in big-O notation.

1. Tiling a $2 \times n$ grid with J, L, and O pieces, where $n \geq 1$ or $n \geq 0$:

```
TetrisDP($n$):
    // your code here
```

2. Counting the number of ways of splitting a string into words:

```
CountSplitDP($S[1, \ldots, n]$):
    // your code here
```

3. Determining if it is possible to buy exactly $n \geq 0$ cupcakes, when Blossom Bake Shop sells them in sets of 6, 9, or 20:

```
CupcakeDP($n$):
    // your code here
```

4. Cutting a piece of wood of size $M \times N$, given a table of prices $P[\,][\,]$:

```
RectDP($P[1, \ldots, M][1, \ldots, N]$):
    // your code here
```