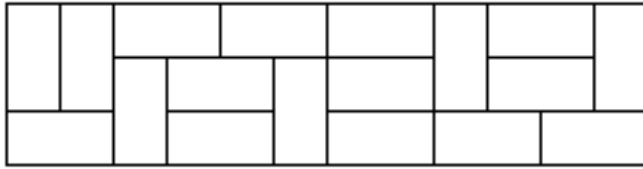


Problem A: Tiling 3-Row Grid

In how many ways can you tile a $3 \times L$ grid with 2×1 tiles?
Here is a sample tiling of a 3×12 rectangle.



Input: length L

Output: the number of possible ways to fully tile the grid.

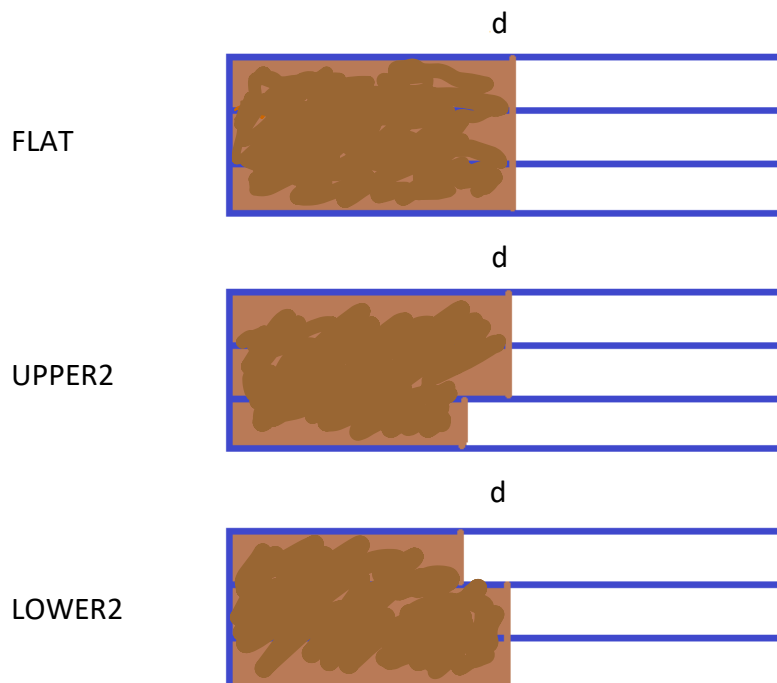
EXAMPLE

INPUT	OUTPUT
2	3
8	153

IMPORTANT NOTE: The objective is **NOT** that you can solve this problem. **Looking for alternative solution online will WASTE YOUR TIME!**

The goal of this class is that you can speed up the provided brute-force algorithm!

- 1) Given that the grid has been fully tiled down to depth d , the problem states may be defined as follows.



Only the tiling that covers the entire length of the grid in FLAT state is the valid complete state.

The possible transitions from **FLAT state** are



The possible transitions from **UPPER2 state** are



Sketch the two possible transitions from LOWER2 state!
Hint: same logic as the UPPER2 state (just flip vertically)

- 2) Based the state transitions in step 1, the recursive brute-force algorithm that solves this problem is implemented as m3tileBF.py (uploaded into “Class Materials” of this week)
- 3) Modify m3tileBF.py into a memoization version m3tileMM.py.
- 4) Modify m3tileMM.py into a dynamic programming version m3tileDP.py. Be certain to determine the correct loop order for d and s!
- 5) Once the memoization and dynamic programming version are verified to be correct, test all three programs and answer the questions below.
 - a) For each program, what is the largest length that takes > 1 second?
 - b) What is the largest length that memoization version reaches its recursion limit?
 - c) What is the advantage of dynamic programming version over the memoization version? What is the reason for such advantage?

PROBLEM B: Shoe-Shopping

This problem is for practicing full process of getting a problem completely solved.

Either memorization or dynamic programming will give the brute-force code enough speed-up to pass all the test cases within the required time limit.

1. Create an account at <https://dmoj.ca> if you have not.
2. Solve this problem: <https://dmoj.ca/problem/dmopc16c1p3>

Hint: Standard process on developing a brute-force solution (all options, what you get from each option and its consequence, recursive solution to the consequence).