HW7

1. Recommended Dynamic Programming Data Structure

The best dynamic programming data structure for MysteryRecursion is a HashMap where:

- The key is a combination of the two input values n and m (e.g., as a string "n,m").
- The value is the result of MysteryRecursion(n, m)

2. Pseudocode for a Memoized Dynamic Programming Algorithm

Input: n: positive integer

Input: m: positive integer

Input: memo: a map to store computed results

```
1 Algorithm: MysteryRecursionMemoized(n, m, memo)
2 if n = 1 and m = 1 then
3     return 1
4 else if (n, m) exists in memo then
5     return memo[(n, m)]
6 else if n = 1 then
7     result ← m · MysteryRecursionMemoized(n, ⌊m/2⌋, memo)
8 else if m = 1 then
9     result ← n · MysteryRecursionMemoized(⌊n/2⌋, m, memo)
10 else
11     result ← n · MysteryRecursionMemoized(⌊n/2⌋, m, memo) +
           m · MysteryRecursionMemoized(n, ⌊m/2⌋, memo)
12 end
13 memo[(n, m)] ← result
14 return result
```

3. Iterative Dynamic Programming Algorithm

Input: n: positive integer

Input: m: positive integer

```
1  Algorithm: MysteryRecursionIterative(n, m)
2  Initialize dp[n+1][m+1] to 0
3  dp[1][1] ← 1
4  for i from 1 to n do
5      for j from 1 to m do
6          if i = 1 and j > 1 then
7              dp[i][j] ← j · dp[i][⌊j/2⌋]
8          else if j = 1 and i > 1 then
9              dp[i][j] ← i · dp[⌊i/2⌋][j]
10         else if i > 1 and j > 1 then
11             dp[i][j] ← i · dp[⌊i/2⌋][j] + j · dp[i][⌊j/2⌋]
12         end
13     end
14 end
15 return dp[n][m]
```