**Questions**

## Question 1:

The # of operations computed in the bottom-up approach is \_\_\_\_, compared to the memoized approach.

1. more
2. less
3. the same

## Question 2:

Why would someone choose the bottom-up approach over the memoized approach?

1. More memory efficient
2. Accessing values is easier
3. a) & b)

## Question 3:

Implement a function named **factorial** that takes in an integer **n** (0 ≤ **n** ≤ 16) and returns the **factorial of n** with the **bottom-up approach**.

Ex.

factorial(5) = 120

## Question 4 breakdown(Extra):

You are given a coordinate plane bounded by (1,1) and (**r**,**c**). Your task is to find the number of possible paths to coordinate (**r**,**c**). However, you are also given **k** cats, each cat is set on a coordinate that you are given and you cannot visit that coordinate.

Your approach from a dynamic programming standpoint:

DP state? (at any index in dp it returns… Ex. dp[i] in the Fibonacci sequence returns ith number)

Base case?

Recurrence?

**Answers and Solutions**

## Question 1:

The answer is c)

Explained in video

## Question 2:

The answer is a)

Explained in video

## Question 3:

Solution: <https://github.com/hamgod420/grade12cs/blob/main/Unit%203%20DSA%20Dynamic%20Programming/Example%20Solutions/FactorialDP.java>

## Question 4:

C++ (Has comments and explanations):

<https://github.com/hamgod420/grade12cs/blob/main/Unit%203%20DSA%20Dynamic%20Programming/Example%20Solutions/ccc12s5.cpp>

Java:

<https://github.com/hamgod420/grade12cs/blob/main/Unit%203%20DSA%20Dynamic%20Programming/Example%20Solutions/ccc12s5.java>