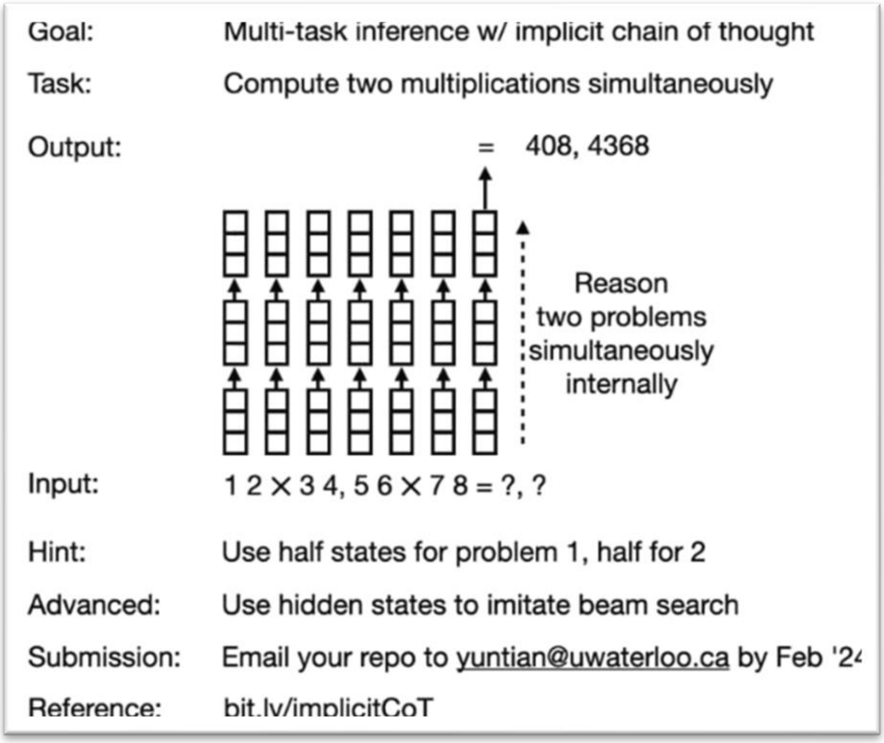
QUESTION:

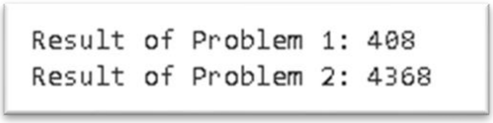


Goal:	Multi-task inference w/ implicit chain of thought
Task:	Compute two multiplications simultaneously
Output:	= 408, 4368
Input:	1 2 × 3 4, 5 6 × 7 8 = ?, ?
Hint:	Use half states for problem 1, half for 2
Advanced:	Use hidden states to imitate beam search
Submission:	Email your repo to yuntian@uwaterloo.ca by Feb '24
Reference:	bit.ly/implicitCoT

CODE:

```
import multiprocessing

def multiply_problem_1(A, B, result_1):
    result_1.value = A * B

def multiply_problem_2(C, D, result_2):
    result_2.value = C * D

if __name__ == '__main__':
    # Sample values for the problems
    A = 12
    B = 34
    C = 56
    D = 78

    result_1 = multiprocessing.Value('i', 0)
    result_2 = multiprocessing.Value('i', 0)

    process_1 = multiprocessing.Process(target=multiply_problem_1, args=(A, B, result_1))
    process_2 = multiprocessing.Process(target=multiply_problem_2, args=(C, D, result_2))

    process_1.start()
    process_2.start()

    process_1.join()
    process_2.join()

    result_problem_1 = result_1.value
    result_problem_2 = result_2.value


    print(f"Result of Problem 1: {result_problem_1}")
    print(f"Result of Problem 2: {result_problem_2}")
```

OUTPUT:



```
Result of Problem 1: 408
Result of Problem 2: 4368
```

Multiprocessing Calculation

Objective:

The objective of this Python script is to demonstrate parallel processing using the **multiprocessing** module to solve two different multiplication problems concurrently.

Code Overview:

The script comprises functions **multiply_problem_1** and **multiply_problem_2**, each designed to perform a specific multiplication task in parallel using separate processes.

Functions:

1. **multiply_problem_1(A, B, result_1)**: Computes the product of integers A and B, storing the result in **result_1**.

2. **multiply_problem_2(C, D, result_2)**: Computes the product of integers C and D, storing the result in **result_2**.

Steps:

1. **Initialization:**
    - Initialize the sample values for the problems:
        - A = 12
        - B = 34
        - C = 56
        - D = 78
    - Create shared memory objects **result_1** and **result_2** using **multiprocessing.Value**.
2. **Process Creation:**
    - Create two separate processes (**process_1** and **process_2**) for each multiplication problem using **multiprocessing.Process**.
    - Each process is targeted to execute the respective multiplication function with its arguments.
3. **Execution:**
    - Start the processes concurrently using **process_1.start()** and **process_2.start()**.
    - Wait for the completion of both processes using **process_1.join()** and **process_2.join()**.
4. **Results Retrieval:**
    - Obtain the computed results from the shared memory locations (**result_1.value** and **result_2.value**).
5. **Output:**
    - Print the results of Problem 1 and Problem 2.

Execution Notes:

- The script showcases the use of multiprocessing to execute independent tasks concurrently, thereby demonstrating the efficiency of parallel processing for computationally intensive operations.
- The results are stored and retrieved using shared memory objects to enable communication between different processes.

Conclusion:

This code demonstrates a basic example of utilizing the **multiprocessing** module in Python to perform parallel computations, exemplifying the distribution of tasks across multiple processes for enhanced efficiency.