Constantino, Raxell Louis I.
J3A

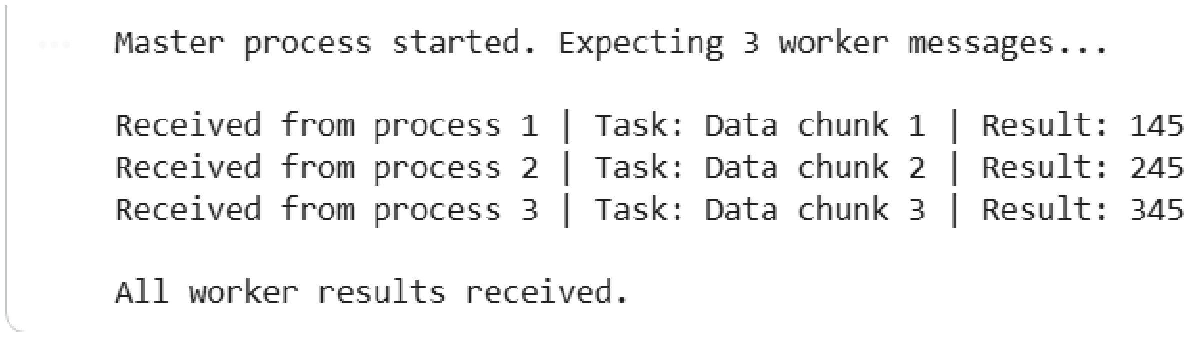
Guide Questions

1. **Why is message passing required in distributed systems?**
- In a distributed system, each process runs on a different machine and has its own memory, so they can't just access each other's data directly. That's why message passing is needed. It lets processes talk to each other by sending and receiving information. For example, in an MPI program, worker processes send their computed results to the master process, which then collects and shows the results. Without message passing, the processes wouldn't be able to work together at all.

2. **What happens if one process fails?**
- If one process stops working in a distributed system, the tasks it was supposed to do might not get done. Other processes that are waiting for messages from it could get stuck, and the whole program might be delayed or incomplete. To handle this, distributed systems can use tricks like reassigning the task to another process, saving progress periodically, or running the same task on multiple processes at the same time. In an MPI program, if a worker fails before sending its result, the master won't receive it, so the output would be missing that part.

3. **How does this model differ from shared-memory programming?**
- Message-passing systems are different from shared-memory programming because, in message passing, each process has its own memory and must communicate explicitly through messages. In shared-memory programming, multiple threads share the same memory, so they can just read and write the same variables. Shared memory needs careful management with locks to avoid conflicts, while message passing is safer for machines that don't share memory. Also, message-passing programs can run on many computers across a network, while shared-memory programs usually run on a single machine.

Code Outputs:

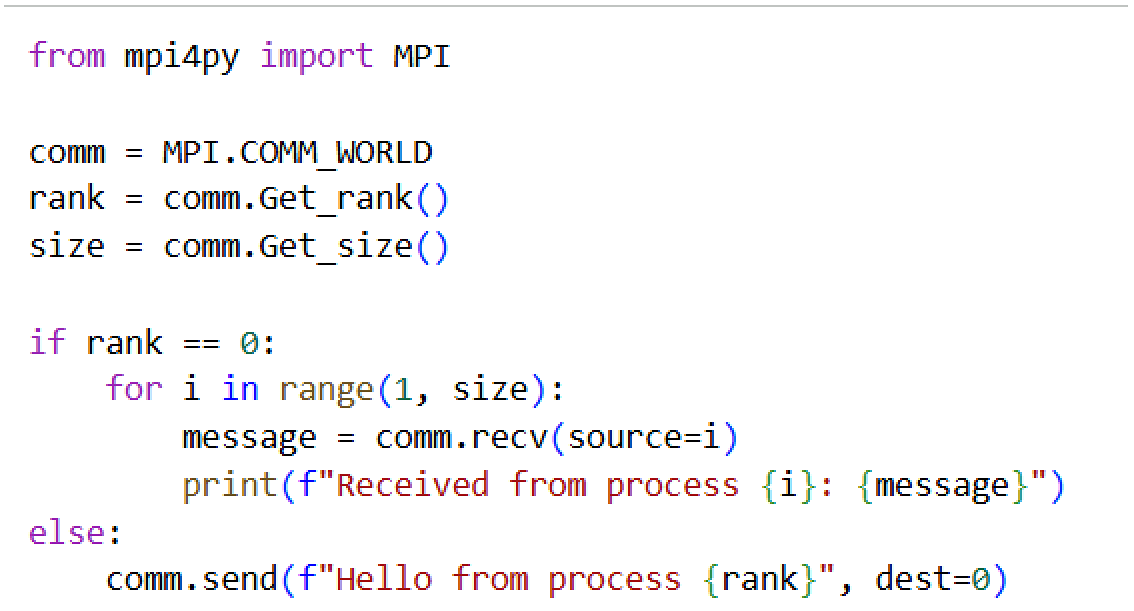


```
Master process started. Expecting 3 worker messages...

Received from process 1 | Task: Data chunk 1 | Result: 145
Received from process 2 | Task: Data chunk 2 | Result: 245
Received from process 3 | Task: Data chunk 3 | Result: 345

All worker results received.
```

Starter code:



```python
from mpi4py import MPI

comm = MPI.COMM_WORLD
rank = comm.Get_rank()
size = comm.Get_size()

if rank == 0:
    for i in range(1, size):
        message = comm.recv(source=i)
        print(f"Received from process {i}: {message}")
else:
    comm.send(f"Hello from process {rank}", dest=0)
```

Modified code:

```
%%writefile mpi_master_worker.py
from mpi4py import MPI

# Initialize MPI
comm = MPI.COMM_WORLD
rank = comm.Get_rank()
size = comm.Get_size()

if rank == 0:
    # Master process
    print(f"Master process started. Expecting {size-1} worker messages...\n")

    for i in range(1, size):
        message = comm.recv(source=i)
        print(
            f"Received from process {message['rank']} | "
            f"Task: {message['task']} | "
            f"Result: {message['result']}"
        )

    print("\nAll worker results received.")

else:
    # Worker processes
    data_chunk = list(range(rank * 10, rank * 10 + 10))
    computed_sum = sum(data_chunk)

    message = {
        "rank": rank,
        "task": f"Data chunk {rank}",
        "result": computed_sum
    }

    comm.send(message, dest=0)
```

```
Writing mpi_master_worker.py
```