Assignment 3

Implement a Distributed Multiplayer Card Game using MPI

Topic: Programming / MPI Application

Submission Requirements

1. A single Python script named: mpi_card_game.py
2. Must run with mpi4py and Python 3
3. Include comments explaining your MPI messaging and logic.
4. Submit a word document with screenshots of the game play and a list of who did what. If your name isn't in the list you get a 0 grade

Overview

Implement a multiplayer distribute card game using Python and MPI (mpi4py). Communication between processes (dealer and players) happens only via MPI.Send and MPI.Recv.

Summary of the game logic

5. Rank 0 acts as the Dealer/Board.
6. All other ranks are Players.
7. The dealer is responsible for shuffling the cards, starting a game and placing a card on the board.
8. Each players take turns matching the rank of the board card, example a 5 or K:
    a. Players will try to match the rank of the board card, if it matches, they play it and it becomes the new board card.
    b. And if there is no match the player passes.
9. The game should continue until a player runs out of cards.

Learning outcomes

1. Understand message passing in MPI.
2. Implement distributed turn-based logic.
3. Handle list operations and state updates across MPI processes.
4. Ensure deadlock-free communication in a distributed application.
5. Apply MPI communication to implement game rules.

Task 1: Setting up the game (10 points)

1. The deck should be a standard 52 card pack. Create a function, createDeck().
2. Shuffle the deck using random.shuffle().
3. Send a hand of 4 or more cards to each player.

Hints:

1. You could use Python lists or tuples: (rank, suit). You could also keep it simple and just create a list of ranks
2. Use comm.send() to send a list of cards to each player.

Task 2: Implement Dealer Logic (35 points)

The dealer (rank 0) must:

1. Start the game by placing a card on the board.
2. Loop over players in round-robin order, sending each their turn signal and the board card.
3. Receive the updated board card and status from each player.
4. Determine if a player has won the game:
    a. If yes, send "win" to all players and terminate gracefully.

Suggestions:

- Each player may receive a hand, a board card and your turn signal/terminating signal. Terminate if someone wins.
- The dealer must not exit prematurely if other ranks are still running.

Task 3: Player Logic (35 points)

Each player (all ranks that are not 0) must:

1. Receive their initial hand from the dealer.
2. When it is their turn, the player should check for a matching card by comparing it with the board card:
    a. If it matches, remove it from the hand. This card should become the new board card.
    b. If no match exists, pass.
    c. Send the updated board card and status ("pass" or "win") to the dealer.
    d. Terminate when a win message is received.
    e. Terminate, if hand is empty and send a win message to the dealer.

Requirements:

1. Only one card can be played per turn.
2. A empty hand means WIN and should be communicated to the dealer for terminating the other ranks.

Task 4: Deadlock Prevention and MPI Termination (15 points)

1. All ranks should terminate gracefully.
2. Do not leave any rank blocked on a recv(). Also check if Barrier() interferes with termination.
3. Avoid sending or receiving extra or unnecessary messages that breaks MPI

Hints:

1. Each send must match a recv exactly, between dealer and player. This is the biggest cause of deadlock.
2. Consider removing unnecessary Barrier() calls.

Task 5: Optional Features (5 points)

For extra credit:

1. Implement multiple rounds until the deck is empty.
2. Add a score system: players gain points for cards played.
3. Allow drawing from deck if a player cannot play.