```go
package main

import (
	"Lab5/models"
	"log"
	"net"
	"net/rpc"
	"os"
)

type Coordinator struct {
	server1 *rpc.Client
	server2 *rpc.Client
}

func (c *Coordinator) TransferMoney(req models.TransactionRequest, res *models.TransactionResponse) error {
	log.Printf("[Coordinator] Starting 2PC transaction: %s → %s, Amount: %d", req.From, req.To, req.Amount)

	fromServer := c.server1
	toServer := c.server2

	if req.From == "account_2" {
		fromServer = c.server2
		toServer = c.server1
	}

	// Phase 1: Prepare
	log.Printf("[Coordinator] Requesting Prepare from %s", req.From)
	err := fromServer.Call("Server.PrepareTransaction", req, res)
	if err ≠ nil || !res.Success {
		log.Printf("[Coordinator] ERROR: Prepare failed for %s, rolling back...", req.From)
		fromServer.Call("Server.RollbackTransaction", req, res)
		return err
	}

	log.Printf("[Coordinator] Requesting Prepare from %s", req.To)
	err = toServer.Call("Server.PrepareTransaction", req, res)
	if err ≠ nil || !res.Success {
		log.Printf("[Coordinator] ERROR: Prepare failed for %s, rolling back...", req.To)
		fromServer.Call("Server.RollbackTransaction", req, res)
		toServer.Call("Server.RollbackTransaction", req, res)
		return err
	}

	// Phase 2: Commit
	log.Printf("[Coordinator] Requesting Commit from %s", req.From)
	err = fromServer.Call("Server.CommitTransaction", req, res)
	if err ≠ nil || !res.Success {
```

```
49                  log.Printf("[Coordinator] ERROR: Commit failed for %s,
    rolling back...", req.From)
50                  toServer.Call("Server.RollbackTransaction", req, res)
51                  return err
52          }
53
54          log.Printf("[Coordinator] Requesting Commit from %s", req.To)
55          err = toServer.Call("Server.CommitTransaction", req, res)
56          if err ≠ nil || !res.Success {
57                  log.Printf("[Coordinator] ERROR: Commit failed for %s,
    rolling back...", req.To)
58                  fromServer.Call("Server.RollbackTransaction", req, res)
59                  return err
60          }
61
62          log.Printf("[Coordinator] SUCCESS: Transaction completed!")
63          return nil
64  }
65
66  func startCoordinator(server1Addr, server2Addr, port string) {
67          server1, err := rpc.Dial("tcp", server1Addr)
68          if err ≠ nil {
69                  log.Fatalf("Failed to connect to server1: %v", err)
70          }
71
72          server2, err := rpc.Dial("tcp", server2Addr)
73          if err ≠ nil {
74                  log.Fatalf("Failed to connect to server2: %v", err)
75          }
76
77          coordinator := &Coordinator{
78                  server1: server1,
79                  server2: server2,
80          }
81
82          rpcServer := rpc.NewServer()
83          rpcServer.Register(coordinator)
84
85          listener, err := net.Listen("tcp", ":"+port)
86          if err ≠ nil {
87                  log.Fatalf("Failed to start coordinator: %v", err)
88          }
89
90          log.Printf("[Coordinator] Running on port %s", port)
91          for {
92                  conn, err := listener.Accept()
93                  if err ≠ nil {
94                          log.Println("[Coordinator] Connection error:", err)
95                          continue
96                  }
97                  go rpcServer.ServeConn(conn)
98          }
99  }
100
101 func main() {
```

```
102         if len(os.Args) < 4 {
103                 log.Fatalf("Usage: go run coordinator.go <server1_addr>
    <server2_addr> <port>")
104         }
105
106         server1Addr := os.Args[1]
107         server2Addr := os.Args[2]
108         port := os.Args[3]
109
110         startCoordinator(server1Addr, server2Addr, port)
111 }
112
```