Name: Shivam Darekar
Prn: 202101040055

Div: C   Batch: C3

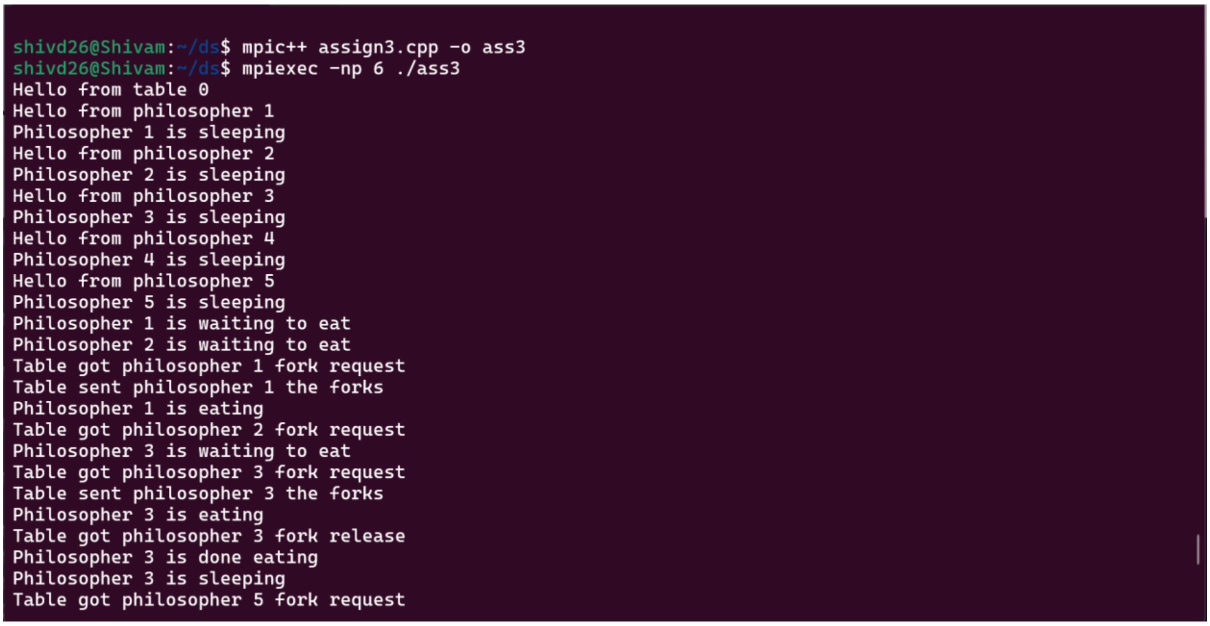
Practical 3

Implement the synchronization for Dining Philosopher Problem using MPI Synchronization Primitives.

Source Code:

```
 #include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <list>

#include <bits/stdc++.h>

#include "mpi.h"


void philosopher(int);

void table(int, int);


#define FORK_REQUEST 1

#define FORK_RESPONSE 2

#define FORK_RELEASE 3

#define DEBUG 1


int main(int argc, char **argv) {

    int myrank, nprocs;


    // Initialize MPI

    MPI_Init(&argc, &argv);

    MPI_Comm_size(MPI_COMM_WORLD, &nprocs);

    MPI_Comm_rank(MPI_COMM_WORLD, &myrank);
    // Depending on rank, assign role of Philosopher or Table
    if (myrank == 0)
        table(myrank, nprocs);
    else
        philosopher(myrank);

    MPI_Finalize();
    return 0;
}

/* Philosopher function - only philosopher processes run this */
void philosopher(int myrank) {
    if (DEBUG)
        printf("Hello from philosopher %d \n", myrank);

    int in_buffer[1];
    int out_buffer[1];
    MPI_Status stat;

    srand(time(NULL) + myrank);

    // Philosopher main loop
    while (true) {
        if (DEBUG)
            printf("Philosopher %d is sleeping \n", myrank);
        sleep(rand() % 10); // Sleep

        if (DEBUG)
            printf("Philosopher %d is waiting to eat \n", myrank);
        MPI_Send(out_buffer, 1, MPI_INT, 0, FORK_REQUEST, MPI_COMM_WORLD); // Request forks
        MPI_Recv(in_buffer, 1, MPI_INT, 0, FORK_RESPONSE, MPI_COMM_WORLD, &stat); // Wait for response
        if (DEBUG)
            printf("Philosopher %d is eating\n", myrank);
        sleep(rand() % 10); // Eat

        if (DEBUG)
            printf("Philosopher %d is done eating \n", myrank);
        MPI_Send(out_buffer, 1, MPI_INT, 0, FORK_RELEASE, MPI_COMM_WORLD); // Release forks
    }
}

/* Table function - only table process runs this */
void table(int myrank, int nprocs) {
    printf("Hello from table %d \n", myrank);

    int in_buffer[1];
    int out_buffer[1];
    int philosopher;
    MPI_Status stat;
    std::list<int> queue;
    bool fork[nprocs - 1];

    // Initialize all forks as free
    for (int i = 0; i < nprocs - 1; i++)
        fork[i] = true;

    // Table main loop
    while (true) {
        MPI_Recv(in_buffer, 1, MPI_INT, MPI_ANY_SOURCE, MPI_ANY_TAG, MPI_COMM_WORLD, &stat); // Receive next message
        philosopher = stat.MPI_SOURCE; // Read source of message
    if (stat.MPI_TAG == FORK_REQUEST) { // Request for forks
      if (DEBUG)
        printf("Table got philosopher %d fork request\n", philosopher);

      if (fork[philosopher % (nprocs - 1)] && fork[philosopher - 1]) {
        // If both forks are free
        fork[philosopher % (nprocs - 1)] = false; // Set forks as taken
        fork[philosopher - 1] = false;
        MPI_Send(out_buffer, 1, MPI_INT, philosopher, FORK_RESPONSE, MPI_COMM_WORLD); //
Send Fork response

        if (DEBUG)
          printf("Table sent philosopher %d the forks\n", philosopher);
      } else {
        // If not both forks are free, add to wait queue
        queue.push_back(philosopher);
      }
    } else if (stat.MPI_TAG == FORK_RELEASE) { // Release of forks
      fork[philosopher % (nprocs - 1)] = true; // Set forks to free again
      fork[philosopher - 1] = true;

      if (DEBUG)
        printf("Table got philosopher %d fork release\n", philosopher);

      // Check if any philosopher in the queue can proceed
      if (!queue.empty()) {
        for (auto it = queue.begin(); it != queue.end();) {
          philosopher = *it;

          if (fork[philosopher % (nprocs - 1)] && fork[philosopher - 1]) {
            // If both forks are free for a philosopher

            fork[philosopher % (nprocs - 1)] = false;

            fork[philosopher - 1] = false;

            MPI_Send(out_buffer, 1, MPI_INT, philosopher, FORK_RESPONSE, MPI_COMM_WORLD); // Send Fork response


            if (DEBUG)

              printf("Table sent philosopher %d the forks\n", philosopher);

            it = queue.erase(it); // Remove from wait list

          } else {

            ++it;

          }

        }

      }

    }

  }

}
```

Output:



```
shivd26@Shivam:~/ds$ mpic++ assign3.cpp -o ass3
shivd26@Shivam:~/ds$ mpiexec -np 6 ./ass3
Hello from table 0
Hello from philosopher 1
Philosopher 1 is sleeping
Hello from philosopher 2
Philosopher 2 is sleeping
Hello from philosopher 3
Philosopher 3 is sleeping
Hello from philosopher 4
Philosopher 4 is sleeping
Hello from philosopher 5
Philosopher 5 is sleeping
Philosopher 1 is waiting to eat
Philosopher 2 is waiting to eat
Table got philosopher 1 fork request
Table sent philosopher 1 the forks
Philosopher 1 is eating
Table got philosopher 2 fork request
Philosopher 3 is waiting to eat
Table got philosopher 3 fork request
Table sent philosopher 3 the forks
Philosopher 3 is eating
Table got philosopher 3 fork release
Philosopher 3 is done eating
Philosopher 3 is sleeping
Table got philosopher 5 fork request
```