Distributed System
Assignment - 4

ASHISH VERMA

20204041

CS - A

Q. Simulate the Distributed Mutual Exclusion.

```
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdbool.h>

#define NUM_PROCESSES 3

int timestamps[NUM_PROCESSES];
bool requesting[NUM_PROCESSES];
bool granting[NUM_PROCESSES];
int critical_section = -1;

void request_critical_section(int process_id) {
   requesting[process_id] = true;
   timestamps[process_id]++;

   for (int i = 0; i < NUM_PROCESSES; i++) {
      if (i != process_id) {
         requesting[i] = true;
         printf("Process %d is requesting access to the critical section from Process %d\n", process_id, i);
         timestamps[process_id] = (timestamps[process_id] > timestamps[i] ? timestamps[process_id] : timestamps[i]) + 1;
         sleep(rand() % 3); // Simulate network delays
         requesting[i] = false;
      }
   }
}

void release_critical_section(int process_id) {
   requesting[process_id] = false;
   for (int i = 0; i < NUM_PROCESSES; i++) {
      if (i != process_id && granting[i]) {
         printf("Process %d is releasing the critical section to Process %d\n", process_id, i);
         granting[i] = false;
      }
   }
   critical_section = -1;
}

void *process_thread(void *arg) {
   int process_id = *((int *)arg);
   while (1) {
      sleep(rand() % 3); // Simulate process activity
      request_critical_section(process_id);

      sleep(rand() % 3); // Simulate process activity
```

```
        critical_section = process_id;
        granting[process_id] = true;
        printf("Process %d is in the critical section\n", process_id);
        sleep(rand() % 3); // Simulate process activity
        release_critical_section(process_id);
    }
    pthread_exit(NULL);
}

int main() {
    srand(time(NULL));
    pthread_t threads[NUM_PROCESSES];
    int process_ids[NUM_PROCESSES];

    for (int i = 0; i < NUM_PROCESSES; i++) {
        process_ids[i] = i;
        pthread_create(&threads[i], NULL, process_thread, &process_ids[i]);
    }

    for (int i = 0; i < NUM_PROCESSES; i++) {
        pthread_join(threads[i], NULL);
    }

    return 0;
}
```

```
Process 0 is requesting access to the critical section from Process 1
Process 0 is requesting access to the critical section from Process 2
Process 0 is in the critical section
Process 1 is requesting access to the critical section from Process 0
Process 2 is requesting access to the critical section from Process 0
Process 2 is requesting access to the critical section from Process 1
Process 1 is requesting access to the critical section from Process 2
Process 2 is in the critical section
Process 2 is releasing the critical section to Process 0
Process 0 is requesting access to the critical section from Process 1
Process 0 is requesting access to the critical section from Process 2
Process 1 is in the critical section
Process 2 is requesting access to the critical section from Process 0
Process 1 is releasing the critical section to Process 2
Process 2 is requesting access to the critical section from Process 1
Process 2 is in the critical section
Process 0 is in the critical section
Process 0 is releasing the critical section to Process 1
Process 0 is releasing the critical section to Process 2
Process 1 is requesting access to the critical section from Process 0
Process 0 is requesting access to the critical section from Process 1
Process 0 is requesting access to the critical section from Process 2
Process 2 is releasing the critical section to Process 0
^C
```