**Assignment 4**

**1.Write a program to simulate logical clock synchronisation using Lamport’s logical clock algorithm and vector clocks.**

**#include <stdio.h>**

**#include <stdlib.h>**

**int max(int a, int b) {**

**return (a > b) ? a : b;**

**}**

**void lamport\_clock(int p1\_events[], int p2\_events[], int n) {**

**int p1\_clock = 0, p2\_clock = 0;**

**printf("Event\tProcess 1\tProcess 2\n");**

**for (int i = 0; i < n; i++) {**

**p1\_clock++;**

**p2\_clock++;**

**if (p1\_events[i] != -1 && p2\_events[i] != -1) {**

**// Communication between processes**

**p1\_clock = max(p1\_clock, p2\_clock) + 1;**

**p2\_clock = max(p1\_clock, p2\_clock) + 1;**

**}**

**printf("E%d\t%d\t\t%d\n", i + 1, p1\_clock, p2\_clock);**

**}**

**}**

**int main() {**

**int p1\_events[] = {1, 0, -1, 1, 0}; // -1 means no communication**

**int p2\_events[] = {0, 1, -1, 1, 0};**

**int n = sizeof(p1\_events) / sizeof(p1\_events[0]);**

**lamport\_clock(p1\_events, p2\_events, n);**

**return 0;**

**}**

**Output:**

Event   Process 1   Process 2

E1      1           2

E2      3           4

E3      4           5

E4      6           7

E5      7           8

**2. Write a program to simulate distributed mutual exclusion algorithm (Lamport and Ricart-agrawala algorithms).**

**Lamport's Algorithm**

**#include <stdio.h>**

**#include <stdlib.h>**

**int lamport\_clock = 0;**

**void send\_request(int process\_id) {**

**lamport\_clock++;**

**printf("Process %d sends request at time %d\n", process\_id, lamport\_clock);**

**}**

**void receive\_request(int process\_id, int sender\_id, int sender\_time) {**

**lamport\_clock = (sender\_time > lamport\_clock) ? sender\_time + 1 : lamport\_clock + 1;**

**printf("Process %d receives request from Process %d at time %d\n", process\_id, sender\_id, lamport\_clock);**

**}**

**void enter\_critical\_section(int process\_id) {**

**printf("Process %d entering critical section at time %d\n", process\_id, lamport\_clock);**

**}**

**void exit\_critical\_section(int process\_id) {**

**printf("Process %d exiting critical section at time %d\n", process\_id, lamport\_clock);**

**}**

**int main() {**

**// Simulating mutual exclusion between two processes**

**int process\_1 = 1;**

**int process\_2 = 2;**

**send\_request(process\_1);**

**receive\_request(process\_2, process\_1, lamport\_clock);**

**enter\_critical\_section(process\_1);**

**exit\_critical\_section(process\_1);**

**send\_request(process\_2);**

**receive\_request(process\_1, process\_2, lamport\_clock);**

**enter\_critical\_section(process\_2);**

**exit\_critical\_section(process\_2);**

**return 0;**

**}**

**Output:**

Process 1 sends request at time 1

Process 2 receives request from Process 1 at time 2

Process 1 entering critical section at time 2

Process 1 exiting critical section at time 2

Process 2 sends request at time 3

Process 1 receives request from Process 2 at time 4

Process 2 entering critical section at time 4

Process 2 exiting critical section at time 4

**Ricart-Agrawala Algorithm**

**#include <stdio.h>**

**#include <stdlib.h>**

**int logical\_clock = 0;**

**void send\_request(int process\_id) {**

**logical\_clock++;**

**printf("Process %d sends request at time %d\n", process\_id, logical\_clock);**

**}**

**void receive\_request(int process\_id, int sender\_id, int sender\_time) {**

**logical\_clock = (sender\_time > logical\_clock) ? sender\_time + 1 : logical\_clock + 1;**

**printf("Process %d receives request from Process %d at time %d\n", process\_id, sender\_id, logical\_clock);**

**}**

**void grant\_permission(int process\_id, int requester\_id) {**

**logical\_clock++;**

**printf("Process %d grants permission to Process %d at time %d\n", process\_id, requester\_id, logical\_clock);**

**}**

**void enter\_critical\_section(int process\_id) {**

**printf("Process %d enters critical section at time %d\n", process\_id, logical\_clock);**

**}**

**void exit\_critical\_section(int process\_id) {**

**printf("Process %d exits critical section at time %d\n", process\_id, logical\_clock);**

**}**

**int main() {**

**// Simulate two processes**

**int process\_1 = 1;**

**int process\_2 = 2;**

**send\_request(process\_1);**

**receive\_request(process\_2, process\_1, logical\_clock);**

**grant\_permission(process\_2, process\_1);**

**enter\_critical\_section(process\_1);**

**exit\_critical\_section(process\_1);**

**send\_request(process\_2);**

**receive\_request(process\_1, process\_2, logical\_clock);**

**grant\_permission(process\_1, process\_2);**

**enter\_critical\_section(process\_2);**

**exit\_critical\_section(process\_2);**

**return 0;**

**}**

**Output:**

Process 1 sends request at time 1

Process 2 receives request from Process 1 at time 2

Process 2 grants permission to Process 1 at time 3

Process 1 enters critical section at time 3

Process 1 exits critical section at time 3

Process 2 sends request at time 4

Process 1 receives request from Process 2 at time 5

Process 1 grants permission to Process 2 at time 6

Process 2 enters critical section at time 6

Process 2 exits critical section at time 6