**Experiment# 2**

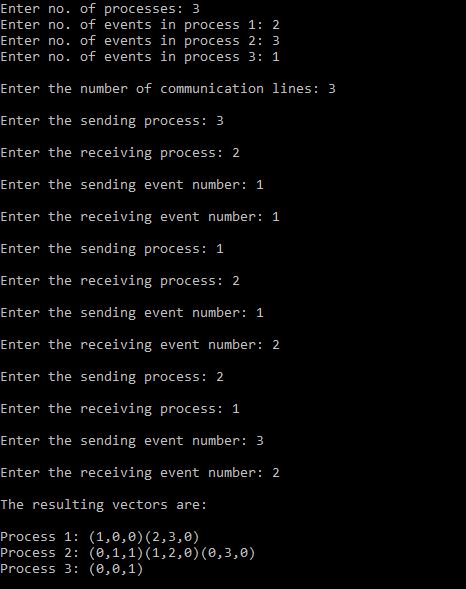
**Experiment Text:** To simulate the functioning of Lamport’s Vector Clock Algorithm

**Outcome:** A vector clock is an algorithm for generating a partial ordering of events in a distributed system and detecting causality violations. Just as in Lamport timestamps, interprocess messages contain the state of the sending process's logical clock. A vector clock of a system of N processes is an array/vector of N logical clocks, one clock per process; a local "smallest possible values" copy of the global clock-array is kept in each process

**Code:**

|  |
| --- |
| #include<iostream> |
|  | #include<conio.h> |
|  | #define SIZE 10 |
|  | using namespace std; |
|  |  |
|  | class node { |
|  | public: |
|  | int data[SIZE]; |
|  | node \*next; |
|  |  |
|  | node() { |
|  | for(int p=0; p<SIZE; p++) { |
|  | data[p] = 0; |
|  | } |
|  | next = NULL; |
|  | } |
|  |  |
|  | node(int v[], int n1) { |
|  | for(int s = 0; s < n1; s++) { |
|  | data[s] = v[s]; |
|  | } |
|  | next = NULL; |
|  | } |
|  |  |
|  | friend class process; |
|  | }\*start=NULL; |
|  |  |
|  | int main() { |
|  | int n, events, sent, receive, sentE, recE, commLines = 0; |
|  | node \*temp; |
|  | node \*proc[SIZE]; //array of processes |
|  | cout<<"Enter no. of processes: "; |
|  | cin>>n; |
|  | int vector[n] = {0}; //representation of data |
|  |  |
|  | /\*----------------INITIALIZATION LOOP-------------------------\*/ |
|  | for(int i = 0; i < n; i++) { //number of processes |
|  | for(int v = 0; v < n; v++) { |
|  | vector[v] = 0; |
|  | } |
|  |  |
|  | cout<<"Enter no. of events in process "<<i+1<<": "; |
|  | cin>>events; |
|  |  |
|  | for(int j = 1; j <= events; j++) { |
|  | vector[i] = j; |
|  | node \*newnode = new node(vector,n); |
|  | if(start == NULL) { |
|  | start = newnode; |
|  | temp = start; |
|  | } else { |
|  | temp->next = newnode; |
|  | temp = temp->next; |
|  | } |
|  | } |
|  | proc[i] = start; |
|  | start = NULL; |
|  | } |
|  |  |
|  | /\*-------------------DATA GATHERING--------------------\*/ |
|  | cout<<"\nEnter the number of communication lines: "; |
|  | cin>>commLines; |
|  | node \*tempS, \*tempR; |
|  |  |
|  | for(int i = 0; i < commLines; i++) { |
|  | cout<<"\nEnter the sending process: "; |
|  | cin>>sent; |
|  | cout<<"\nEnter the receiving process: "; |
|  | cin>>receive; |
|  | cout<<"\nEnter the sending event number: "; |
|  | cin>>sentE; |
|  | cout<<"\nEnter the receiving event number: "; |
|  | cin>>recE; |
|  |  |
|  | tempS = proc[sent - 1]; |
|  | tempR = proc[receive - 1]; |
|  |  |
|  | for(int j = 1; j < sentE; j++) |
|  | tempS = tempS->next; |
|  |  |
|  | for(int j = 1; j < recE; j++) |
|  | tempR = tempR->next; |
|  |  |
|  | for(int j = 0; j < n; j++) { |
|  | tempR->data[j] = (tempR->data[j] < tempS->data[j]) ? tempS->data[j] : tempR->data[j]; |
|  | } |
|  | } |
|  |  |
|  | /\*-------------------DISPLAYING------------------------\*/ |
|  | cout<<"\nThe resulting vectors are:\n\n"; |
|  | for(int k = 0; k < n; k++) { |
|  | cout<<"Process "<<k + 1<<": "; |
|  |  |
|  | node \*temp1 = proc[k]; |
|  | while(temp1) { |
|  | cout<<"("; |
|  | for(int f = 0; f < n - 1; f++) |
|  | cout<<temp1->data[f]<<","; |
|  |  |
|  | cout<<temp1->data[n-1]; |
|  | cout<<")"; |
|  | temp1 = temp1->next; |
|  | } |
|  | cout<<endl; |
|  | } |
|  |  |
|  | return 0; |
|  | } |

**Expected Outcome:**

****