**18CS621: Distributed Systems**

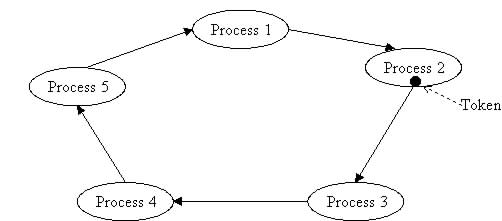
**Lab Assignment: Coordination**

**Date: - 28/11/2020**

**Individual Contributions to Assignment**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Name** | **Contributions** |
| 20010 | C H Vamshi | Mutual Exclusion (Token Ring) |
| 20017 | K Koushik | Leader Election (Ring Election) |
| 20031 | 1. Aathith Bhalaje | Vector Clocks Synchronization |

**Token Ring Mutual Exclusion: -**



**How the Algorithm Works: -**

* All the processes are arranged in a Ring like format
* Initially a Token is given to a process inside the ring. The token rotates in the ring among all processes one at a time, if that process wants to enter Critical Section it carries the token and gets accessed into Critical Section. If it does not want then it passes on that token to other process which is next to it in the ring.
* If a process wants to enter Critical Section and does not have the token, it waits until the token is been passed to that process and when it receives it enters Critical Section executes and passes on the token to other process in the ring

**How we tried to Implement: -**

* Here we assumed each process to be a Personal Computer which are connected to a same Printer or Xerox Machine.
* So, the Xerox Machine can be considered as the Critical Section
* Initially a token is assigned to a random Computer
* Now the user is asked if he wants to access Xerox machine from a particular computer
* Since the token is passed from one Computer to other in the Ring, whenever the PC which we gave the task to access the printer gets the token it utilizes the Printer (Xerox Machine) and passes the Token

**Instructions to Run Code: -**

* **Files Structure: -**

-Co-Ordination Assignment

|-Problem1(Vector Clock)

|- vectorClock.go

|- VectorClock.docx (Documentation)

|-Problem3(Mutual Exclusion)

|-tokenRingMuTex.go

|-Token Ring Mutual Exclusion.docx (Documentation)

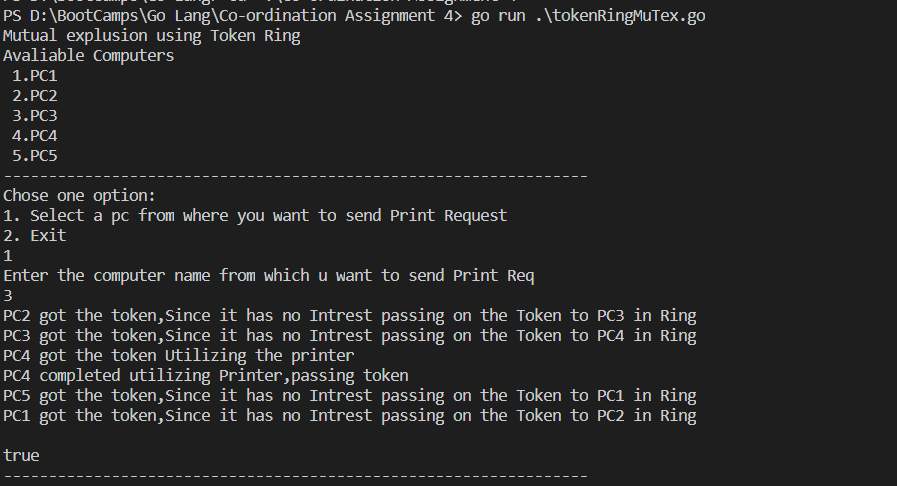
|-Problem4(Leader Election)

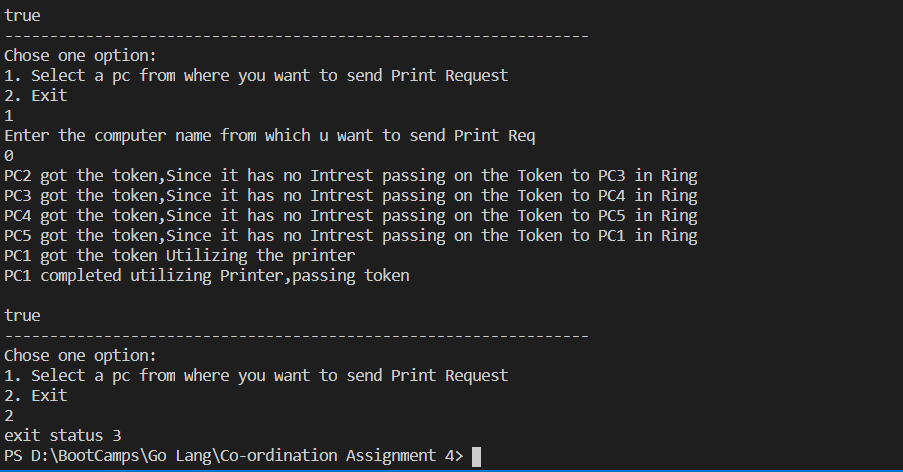
|-ringLeaderElection.go

|-Election Leader.docx (Documentation)

* **How to give Inputs: -**
  + This is a Menu Driven Application
  + When the Application starts the user is provided with two options (1 or 2)
  + Enter 1 or 2 into Terminal
  + Selecting 2 exits the application
  + Upon Selecting 1 you are prompted to enter the PC number (0 to 5) from which you wanted to generate a request to Printer
    - 0 -> PC1
    - 1 -> PC2
    - 2 ->PC3
    - 3 ->PC4
    - 4 ->PC5
  + It shows you how the Token is Passed in Ring and the request is served

**Output Screenshots: -**





Cons of Token Ring: -

If a PC wants to access Printer it waits until it gets the token

If it is last in the ring while serving the token it must wait until all the process pass it among themselves even, when they have no use of it

So, a starvation condition might occur