Program to implement the Ricart-Agrawala algorithm for implementing distributed mutual exclusion.

System Requirements:

1. **Operating System: Windows 10**
2. **Python should be installed in your PC**
    a. Open Command Prompt > Type **Python**, Hit Enter If Python Is Installed it will show the version Details
    b. Python 2 or Python 3 should be installed.
    c. **'threading'** and **'sys'** packages should be part of your python installation. Generally, it comes by default.

Instructions to execute the code:

1. Extract the contents of zip folder in a new folder.
2. To start the execution, you have have 2 options:
    a. Rename "**RA_run.txt**" to "**RA_run.bat**" and Execute the file named as "**RA_run.bat**" by double clicking on it.
    b. Or execute the command "**python RA_algo.py"** in a Command Prompt at the location where these files are stored.
3. Then you will be asked to enter the number of sites which need to be simulated.

    Eg:

    ```
    Enter 0 for working with default values
    Enter the number of sites :4
    ```

    You can enter **0,** if you would like to simulate with default values in code. Then, and simulation will start printing the logs from each site, please jump to step 6.

    For each site a separate **thread** will be created. Minimum of 2 sites will be used in code.

4. Then you will be asked to enter the number of events in each site.

    To make the data entry, easy it assumed that the number of events are same in each site. So it is recommended to enter the number of events as the maximum events in one site.
    Eg:

    ```
    Enter 0 for working with default values
    Enter the number of sites :4
    Enter the number events in each site :6
    ```

5. Then you will be asked to enter the critical events in each site, one be one:

```
Enter 0 for working with default values
Enter the number of sites :4
Enter the number events in each site :6
Enter the index of critical events in site 1 seperated by space
2 3
Enter the index of critical events in site 2 seperated by space
4 5
Enter the index of critical events in site 3 seperated by space
5
Enter the index of critical events in site 4 seperated by space
3
```

6. Once the data entry is completed, the simulation parameters are printed and output starts printing the logs from all sites.

   Eg:

```
number of sites :4
number events in each site :6
 critical events in each site :
[[2, 3], [4, 5], [5], [3]]
```

7. How to interpret the logs:

   a. Different indentation is given to print from different to sites, to differentiate it easily: An example output is given below. Site id in different colors to make the indentation clear.

```
site 1 started
        site 2 started
                site 3 started

                        site 4 started
                site 3 : event 1 start with (ts,pi)=(1,3)
        site 2 : event 1 start with (ts,pi)=(1,2)
site 1 : event 1 start with (ts,pi)=(1,1)
```

   b. Need for critical section, requests replys, started of critical section etc will be printed as shown below.

```
site 1 : event 2 with (ts,pi)=(2,1)need CS entry

site 1 : requested CS to site 2 with (ts,pi)= (2,1)
        site 2 : replied to site 1 with timestamp 2 for request(2,1)->(2,2)


site 1 : requested CS to site 3 with (ts,pi)= (2,1)
                site 3 : replied to site 1 with timestamp 2 for request(2,1)->(2,3)


site 1 : requested CS to site 4 with (ts,pi)= (2,1)
                        site 4 : replied to site 1 with timestamp 2 for request(2,1)->(2,4)


site 1 : event with (ts,pi)=(2,1)received all replys

site 1 : event 2 ___critical section__ start
```

   c. Once the critical section is over, Request Deferred array is also printed by the same site.

```
site 1 : event 2 ___critical section__ end

                site 3 : event 3 start with (ts,pi)=(3,3)

                                site 4 : requested CS to site 3 with (ts,pi)= (3,4)

site 1 : Request_Deferred : [0, 0, 0, 1]

                site 3 : replied to site 4 with timestamp 3 for request(3,4)->(3,3)
```

d. Once each site finishes its events, it waits for other sites to finish and prints this info. Once all sites are finished the algorithm terminates:

```
                site 3 is waiting for other sites to finish
        site 2 is stopping...
site 1 is stopping...
                site 3 is stopping...

Done!
```