Assignment 8

Operating System Lab (CS342)

Department of CSE, IIT Patna

Date:- 31-March-2020 **Time**:- 3 hours

Instructions:

1. All the assignments should be completed and uploaded by 5.30 pm. Marks will be deducted for submissions made after 5.30 pm.
2. Markings will be based on the correctness and soundness of the outputs. Marks will be deducted in case of plagiarism.
3. Proper indentation and appropriate comments (if necessary) are mandatory.
4. You should zip all the required files with file names **Q1.c, Q2.c, Q3.c** and name the zip folder as **roll_no.zip, eg. 1701cs11.zip**.
5. Upload the assignments (the zipped folder) in the following link: https://www.dropbox.com/request/H7VSIjoivUzjYJvhePQg

Questions:

1. **WRITE A C PROGRAM TO SIMULATE ALGORITHM FOR DEADLOCK PREVENTION: -**
    - Start the program
    - Attacking Mutex condition: never grant exclusive access. But this may not be possible for several resources.
    - Attacking pre-emption: not something you want to do.
    - Attacking hold and wait condition: make a process hold at the most 1 resource at a time. Make all the requests at the beginning. Nothing policy. If you feel, retry.
    - Attacking circular wait: Order all the resources. Make sure that the requests are issued in the correct order so that there are no cycles present in the resource graph. Resources numbered 1 ... n. Resources can be requested only in increasing order. i.e. you cannot request a resource whose no is less than any you may be holding.
    - Stop the program

- Sample Output

  SIMULATION OF DEADLOCK PREVENTION

```
Enter no. of processes, resources 3, 2
Enter allocation matrix
2 4 5
3 4 5
Enter max matrix
4 3 4
5 6 1
Enter available matrix
2
Failing: Mutual Exclusion
By allocating required resources to process deadlock is prevented
Lack of no pre-emption deadlock is prevented by allocating needed
resources
Failing: Hold and Wait condition
```

2. WRITE A C PROGRAM TO SIMULATE ALGORITHM FOR DEADLOCK DETECTION: -

```
1) Let Work and Finish be vectors of length ‘m’ and ‘n’ respectively.
Initialize: Work = Available
Finish[i] = false; for i=1, 2, 3, 4….n

2) Find an i such that both
a) Finish[i] = false
b) Need$_i$ <= Work
if no such i exists goto step (4)
3) Work = Work + Allocation[i]
Finish[i] = true
goto step (2)

4) if Finish [i] = true for all i
then the system is in a safe state
  Else Deadlock Detected.
```

- Sample Output

```
Enter the no of process: 4
Enter the no of resources: 5

Total Amount of the Resource R1: 2
Total Amount of the Resource R2: 1
Total Amount of the Resource R3: 1
Total Amount of the Resource R4: 2
Total Amount of the Resource R5: 1
```

```
Enter the request matrix:
0 1 0 0 1
0 0 1 0 1
0 0 0 0 1
1 0 1 0 1

Enter the allocation matrix:
1 0 1 1 0
1 1 0 0 0
0 0 0 1 0
0 0 0 0 0
Deadlock detected
--------------------------------
```

3. WRITE A C PROGRAM TO SIMULATE ALGORITHM FOR DEADLOCK AVOIDANCE (BANKER'S ALGORITHM) AND PRINT ALL POSSIBLE SAFE SEQUENCES.

- **Sample Output**

```
Enter the no of process: 5
Enter the no of resources: 3

Total Amount of the Resource R1: 3
Total Amount of the Resource R2: 3
Total Amount of the Resource R3: 2

Enter the max matrix:
P0 7 5 3
P1 3 2 2
P2 9 0 2
P3 2 2 2
P4 4 3 3
Enter the allocation matrix:
P0 0 1 0
P1 2 0 0
P2 3 0 2
P3 2 1 1
P4 0 0 2
Safe sequence 1: P1 -> P3 -> P4 -> P0 -> P2
Safe sequence 2: P1-> P4-> P0-> P2-> P3
```