BCSE – 3rd year – 1st Semester – 2023
Operating Systems Laboratory
Assignment – II [Q1-Q4: CO2, Q5-Q7: CO3, Q8: CO4]

1. Create child processes: X and Y. a. Each child process performs 10 iterations. The child process displays its name/id and the current iteration number, and sleeps for some random amount of time. Adjust the sleeping duration of the processes to have different outputs (i.e. another interleaving of processes' traces).
   b. Modify the program so that X is not allowed to start iteration *i* before process Y has terminated its own iteration *i-1*. Use semaphore to implement this synchronization.
   c. Modify the program so that X and Y now perform in lockstep [both perform iteration I, then iteration i+1, and so on] with the condition mentioned in Q (2b) above.
   d. Add another child process Z.
   Perform the operations as mentioned in Q (1a) for all three children.
   Then perform the operations as mentioned in Q (1c) [that is, 3 children in lockstep].

2. Write an IPC program to send and receive message (between a child and her mother) using *Message Queue*.
3. Write an IPC program to send and receive message (between a child and her mother) using *Pipe*.
4. Write an IPC program using *shared memory* to share notes between a mother and her child.
5. Write a program for *p*-producer *c*-consumer problem. A shared circular buffer that can hold 20 items is to be used. Each producer process can store any numbers between 1 to 50 (along with the producer id) in the buffer. Each consumer process can read from the buffer and add them to a variable GRAND (initialized to 0). Though any consumer process can read any of the numbers in the buffer, the only constraint being that any number written by some producer should be read exactly once by exactly one of the consumers.
(a) Assume 5 *producers* and 10 *consumers*, with each *producer* doing 10 iterations and each *consumer* doing 4 iterations. .
(b) After the rounds are finished, the parent process prints the value of GRAND.
(c) Can you induce *race condition* in this problem? Justify your answer.
6. Write a program for the Reader-Writer process for the following situations:
a) Multiple readers and one writer: writer gets to write whenever it is ready (reader/s wait)
b) Multiple readers and multiple writers: readers get priority over any writer (writer/s wait)
c) Multiple readers and multiple writers: any writer gets to write whenever it is ready, provided no other writer is currently writing (reader/s wait)
7. Implement Dining Philosophers' problem using Monitor. Test the program with (a) 5 philosophers and 5 chopsticks, (b) 6 philosophers and 6 chopsticks, and (c) 7 philosophers and 7 chopsticks
8. Design a CPU scheduler for jobs whose execution profiles will be in a file that is to be read and appropriate scheduling algorithm to be chosen by the scheduler.

   Format of the profile:
   <Job id> <priority> <arrival time> <CPU burst(1) I/O burst(1) CPU burst(2) ……. >-1
   (Each information is separated by blank space and each job profile ends with -1. Lesser priority number denotes higher priority process with priority number 1 being the process with highest priority.)
   Example: 2 3 0 100 2 200 3 25 -1 1 1 4 60 10 ….. -1 etc.
   Testing:
   a. Create job profiles for 20 jobs and use the following scheduling algorithms (Priority and Round Robin (time slice:15)).
   b. Compare the average waiting time, turnaround time of each process for the different scheduling algorithms.

Last Date for submission and viva: Respective groups: September 25th / 27th /28th 2023