ASSIGNMENT 6

Title : Deadlock Avoidance using Semaphores

Problem Statement : Implement the deadlock-free solution to Dining Philosophers problem to illustrate the problem of deadlock and/or starvation that can occur when many synchronized threads are competing for limited resources.

Theory :

Deadlock - a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process.

Deadlock can arise if the following 4 conditions hold simultaneously:

- Mutual Exclusion - One or more than one resource are non-sharable (Only one process can use at a time)
- Hold and Wait - A process is holding atleast one resource and waiting for resources.
- No Preemption - A resource cannot be taken from a process unless the process releases the resource.
- Circular Wait - A set of processes are waiting for each other in circular form

Starvation is the problem that occurs when high priority processes keep executing and low priority processes get blocked for indefinite time. In heavily loaded computer system, a steady stream of higher-priority processes can prevent a low-priority process from ever getting the CPU. In starvation resources are continuously utilised by high priority processes. Problem of starvation can be resolved using Aging. In Aging priority of long waiting processes is gradually increased.

The Dining Philosopher problem states that K philosophers seated around a circular table with one chopstick between each pair of philosopher. There is one chopstick between each philosopher. A philosopher may eat if he can pickup the two chopstics adjacent to him. One chopstick may be picked up by any one of its adjacent followers but not both.

Algorithm (semaphore solution)

```
process P[i]            // for each philosopher
while true do
{
      THINK;
      PICKUP (CHOPSTICK[i], CHOPSTICK [i+1 mod 5]);
      EAT;
      PUTDOWN (CHOPSTICK [i], CHOPSTICK [P+1 mod 5]);
}
```

Conclusion: I have successfully implemented the deadlock - free solution to Dining Philosophers problem to illustrate the problem of deadlock and / or starvation that can occur when many synchronized threads are competing for limited resources.