# **Operating Systems**

# **LAB 05**

**Objective: Understand Deadlock Avoidance/Prevention Techniques**

|  |
| --- |
| **Excercise 1 - Deadlock** |
| In a typical producer/consumer secinario, if you are using a buffer, a producer and a consumer.  Assume you are using   * One semaphore (**mutex**) to prevent race condition on the buffer. * One semaphore (**full**) to maintain produced items in the buffer * One semaphore (**empty**) to maintain empty places in the buffer   The contents of the Produce and Consume functions are organized as shown below.   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Produce()** | |  | **Consume()** | | | **wait(mutex)**  **wait(empty)**  **PRODUCE()**  **signal(full)**  **signal(mutex)** | // mutex--  // empty--  // full++  // mutex++ |  | **wait(mutex)**  **wait(full)**  **CONSUME()**  **signal(empty)**  **signal(mutex)** | // mutex--  // empty--  // full++  // mutex++ | |
| * **Describe a scenario where a deadlock could happen.** * **Rearrage the code so as to prevent deadlock** |

**Exercise 2. Write a c program that simulates “Banker’s Algorithm”**