Lecture 06 Assignments

1. What is a *semaphore*? Complete the semaphore solution of the **bounded buffer producer-consumer** problem by filling the blanks of the given *producer* and *consumer* structures:

```
A bounded buffer with n locations;

Semaphore mutex initialized to the value 1;

Semaphore full initialized to the value 0;

Semaphore empty initialized to the value n;
```

Producer process

```
do  {     produce an Item;

      1.1 ----------------------------------

      1.2 ----------------------------------

            add item to the buffer;   //Critical Section

      1.3 ----------------------------------

          signal (full); // signal to consumer that the buffer is full

} while (TRUE);
```

Consumer process

```
do {

      1.4 ----------------------------------

      1.5 ----------------------------------

            remove an item from the buffer; //Critical Section

            signal (mutex); // signal to producer that the  mutex is free

      1.6 ----------------------------------

      } while (TRUE);
```

2). Complete the semaphore solution of *Readers priority situation* of **Readers-Writers** synchronization problem by filling in the blanks of the structure of the *Readers process* based on the following data:

i. Dataset

ii. Semaphore **rw_mutex** initialized to 1

iii. Semaphore **mutex** initialized to 1

iv. Integer **read_count** initialized to 0

Readers process

```
do  {

    //first finding readers using mutex

2.1 ----------------------------
    read_count++;  // find readers

    if (read_count == 1) //if at least one reader

    wait(rw_mutex); // then a writer should wait

2.2 ----------------------------

    /* reading is performed */

2.3 ----------------------------

      read count--;  // reading by readers
    if (read_count == 0)  // if no more readers

2.4 ----------------------------

    signal(mutex);  // signal 'mutex' to synchronized writers

} while (TRUE);
```

3). What is a Critical Section (CS)? How would the *semaphore* solve the issue(s) of the CS in a process synchronization problem?

4). How does an OS recognize the user-level threads for its execution? Describe the importance of **Light Weight Processes (LWPs)** in this scenario.

5). What are the two Inter-Process Communication (IPC) models? What are the strengths and weaknesses of the two approaches?

6). Identify the nature of the process structure, including its IPC model, which is shown below:

```
while (true)
    {
        if (counter == BUFFER_SIZE)
           /* do nothing */
        Buffer[i] = next_item;
        in = (in + 1) % BUFFER_SIZE;
        counter++;  }
```

7). The execution sequence of producer and consumer processes in a multiprogramming system is shown in the **Figure** below (where the counter is a shared variable for both the producer and consumer processes). Based on the figure, answer the following:

7.1) Check whether any **race condition** occurs in the interleaved execution of processes. Describe its reason(s).

7.2) Is this a process synchronization problem? Why?

Time	Process	Register-counter Status	Value
T_0	producer	$register_1$ = counter	$register_1 = 5$
T_1	producer	$register_1$ += 1	$register_1 = 6$
T_2	producer	counter = $register_1$	counter = 6
T_3	consumer	$register_2$ = counter	$register_2 = 6$
T_4	consumer	$register_2$ −= 1	$register_2 = 5$
T_5	consumer	counter = $register_2$	counter = 5
T_6	producer	$register_1$ = counter	counter = 5

8). What is the meaning of the term **busy waiting**? What other kinds of waiting are there in an OS? Can busy waiting be avoided altogether? Explain your answer.

9). Show that if the **wait()** and **signal()** semaphore operations are not executed atomically, then mutual exclusion may be violated.

10). The implementation of **mutex locks** suffers from **busy waiting**. Describe what changes would be necessary so that a process waiting to acquire a **mutex lock** would be blocked and placed into a waiting queue until the lock became available.