Assignment -2

1. What is synchronization in the context of operating systems, and mention the problems that could occur if synchronization is not achieved.
2. What is a critical section? Why is it necessary to protect critical sections in a concurrent environment?
3. What is a race condition? Explain using the example of the Producer–Consumer problem.
4. List and define the three conditions needed to achieve synchronization.
5. Explain the concept of strict alternation as a software solution for synchronization. Under what conditions is it effective?
6. Discuss Peterson's solution and how it ensures mutual exclusion, progress and bounded waiting.
7. What is semaphore? Explain the types of the semaphore.
8. Describe Down()/ P() and Up()/V() operations for counting and binary semaphores.
9. Explain how semaphores can be used to solve the producer-consumer problem.
10. Present the dining philosopher problem. Why does it illustrate challenges in synchronization? Provide a solution to the dining philosopher problem using semaphores.
11. Define the reader-writer problem. Explain how semaphores can be used to address the reader-writer problem.
12. Define a monitor in the context of synchronization. Provide an example of producer–consumer problem and explain how monitors can be applied to solve a synchronization problem.
13. In the below reader-write implementation using semaphore, consider the following situations and answer accordingly.

```
typedef int semaphore;              /* use your imagination */
semaphore mutex = 1;                /* controls access to rc */
semaphore db = 1;                   /* controls access to the database */
int rc = 0;                         /* # of processes reading or wanting to */

void reader(void)
{
      while (TRUE) {                /* repeat forever */
            down(&mutex);           /* get exclusive access to rc */
            rc = rc + 1;            /* one reader more now */
            if (rc == 1) down(&db); /* if this is the first reader ... */
            up(&mutex);             /* release exclusive access to rc */
            read_data_base();       /* access the data */
            down(&mutex);           /* get exclusive access to rc */
            rc = rc – 1;            /* one reader fewer now */
            if (rc == 0) up(&db);   /* if this is the last reader ... */
            up(&mutex);             /* release exclusive access to rc */
            use_data_read();        /* noncritical region */
      }
}


void writer(void)
{
      while (TRUE) {                /* repeat forever */
            think_up_data();        /* noncritical region */
            down(&db);              /* get exclusive access */
            write_data_base();      /* update the data */
            up(&db);                /* release exclusive access */
      }
}
```

(A) What happens, if we interchange, Down(&mutex) and rc = rc+1 in the reader's code.

(B) What happens if we interchange if(rc==1) down(&db) and up(mutex) in reader's code.

14. Suppose we want to synchronize two concurrent processes P and Q using binary semaphores S and T. The code for the processes P and Q is shown below.

```
Process P:
while (1) {
W:
   print '0';
   print '0';
X:
}

Process Q:
while (1) {
Y:
   print '1';
   print '1';
Z:
}
```

Synchronization statements can be inserted only at points W, X, Y and Z.

Which of the following will ensure that the output string never contains a substring of the form 01^n0 or 10^n1 where n is odd? **Justify your answer with analysis.**

(A) P(S) at W, V(S) at X, P(T) at Y, V(T) at Z, S and T initially 1
(B) P(S) at W, V(T) at X, P(T) at Y, V(S) at Z, S and T initially 1
(C) P(S) at W, V(S) at X, P(S) at Y, V(S) at Z, S initially 1
(D) V(S) at W, V(T) at X, P(S) at Y, P(T) at Z, S and T initially 1