1. A _____process is one that can affect or be affected by other processes executing in the system
2. __ share the logical address space
3. Concurrent access to shared data may result in _____
4. Concurrent processes run on uni-processor systems with the help of __ and parallel processes on multi-processor systems with the help of __
    a. Interleaving, Overlapping
    b. Overlapping, Interleaving
5. while (counter == BUFFER SIZE); - This loop an example for __ state of a process
6. Producer and Consumer

```
while (true) {
     /* produce an item in next_produced */

     while (counter == BUFFER_SIZE)
        ; /* do nothing */

     buffer[in] = next_produced;
     in = (in + 1) % BUFFER_SIZE;
     counter++;
}
```

```
while (true) {
     while (counter == 0)
        ; /* do nothing */

     next_consumed = buffer[out];
     out = (out + 1) % BUFFER_SIZE;
     counter--;

     /* consume the item in next_consumed */
}
```

   Imagine that the counter variable used in the producer-consumer problem is initially 5and it becomes 4 after the execution of Producer and Consumer. Illustrate the sequence of execution that would have taken place during the modification of the counter which would have resulted in the value being changed to 4
7. A situation like this, where several processes access and manipulate the same data concurrently and the outcome of the execution depends on the particular order in which the access takes place, is called a_____
8. Which one of the following is incorrect about critical section ?
    a. Processes ,ay be changing common variables, updating a table, writing a file, and so on inside the CS
    b. CS contains the collection of statements that meant to manipulate the critical resource
    c. If a process has entered into the CS for accessing Resource then no other process can enter into the CS for accessing any resource
    d. Processes may be concurrently executing their remainder sections without mutual exclusion
9. Name the three requirements to be satisfied by the solutions developed for CS problem
10. Let's say P1 wants to enter its CS. Even though no process is inside that CS, P1 is denied entry. Which one of the requirements is not fulfilled?
11. Which kernels are free from race conditions and why?
12. Mention the drawbacks of Peterson's solution
13. Identify the mistake if any in the following algorithm. Which one of the requirements won't be fulfilled due to that mistake?

```
do{
    flag[i]=true;
    turn = j;
    while( flag[j] && turn != j);
       CS
    flag[i] = false;

while(true);
```

14. Name the hardware instructions used for the critical section problem
15. What is the drawback of mutex locks ?

16. Mutex lock is also called a _____
17. Name the components of a semaphore
18. Find the mistake if any

```
wait(semaphore *S) {
        S->value--;
        if (S->value <= 0) {
                add this process to S->list;
                block();
        }
}
```

19. Consider that the value of Semaphore is -4. Then how many processes are blocked and how many are inside the CS at the moment
20. Name the types of semaphores
21. Whether deadlock is possible among the following processes ?

P_0	P_1
wait(S);	wait(S);
wait(Q);	wait(Q);
.	.
.	.
.	.
signal(S);	signal(Q);
signal(Q);	signal(S);

22. We have three processes—L, M, and H —whose priorities follow the order L < M < H. Assume that process H requires resource R, which is currently being accessed by process L. So H waits for H to release R. Suppose that process M becomes runnable, thereby preempting process L. This situation leads to the problem of ___
23. True or false: Mutual exclusion eliminates deadlocks.
24. Name the semaphores used in the bounded buffer problem with their initial values
25. In a reader-writer problem, ME is not needed among___
26. Fill in the blanks in the algorithm

```
do {
   wait(mutex);
   read_count++;
   if (read_count ==__)
      wait(rw_mutex);
   signal(mutex);
      . . .
   /* reading is performed */
      . . .
   wait(mutex);
   read_count--;
   if (read_count ==__)
      signal(rw_mutex);
   signal(mutex);
} while (true);
```