CSE 313 Operating Systems

Class Test 1 Marks: 20 Time: 20 Minutes

Name _____ **Student ID** _____

There are **four** questions. Answer ALL.

1. **[6 points]** Consider the following solution for the producer-consumer problem. Give an example of a scenario where this code fails. Assume down and up operations have their standard definition.

```
#define N = 100
typedef int semaphore;
semaphore mutex = 1;
int count = 0;

void producer (void)
{
   int item;
   while (TRUE){
         item = produce();
         down(&mutex);
         if (count == N)
            sleep();
         inset(item);
         count = count + 1;
         if (count == 1)
            wakeup(consumer);
         up(&mutex);
   }
}
```

```
void consumer (void)
{
   int item;
   while (TRUE){
         down(&mutex);
         if (count == 0) sleep();
         item = remove-item();
         count = count - 1;
         if (count == N-1)
            wakeup(producer);
         up(&mutex);
        consume(item);

   }
}
```

Answer:

When N ==1
Producer enters the critical region by down(mutex).
Now since count = N, producer will call sleep();

On the other hand, consumer will try to enter the critical region by calling down (mutex), but it cannot. So it will sleep too.

Both will sleep forever.

2. **[4 points]** Now consider yet another solution for the producer-consumer problem that eliminates the failure scenario that is present in the solution in Question No 1. Give an example scenario to show the demerit of this solution.

```
#define N = 100
typedef int semaphore;
semaphore mutex = 1
int count = 0;
void producer (void)
{  int item;
   while (TRUE){
         item = produce();
         down(&mutex);
         if (count < N) {
               inset(item);
               count = count + 1;}
         up(&mutex);
   }
}
```

```
void consumer (void)
{
   int item;
   while (TRUE){
         down(&mutex);
         if (count > 0) {
               item =    remove();
               count = count - 1;}
         up(&mutex);
         consume(item);

   }
}
```

Answer:

Busy waiting problem.
Wastage of resources.

Suppose when count = N, the producer cannot insert an item. Suppose the scheduler does not schedule consumer for a long period of time.. In the meantime, producer will unnecessarily execute the loop utilizing the CPU.

3. **[6 points]** Consider you have two processes called Examiner and Scrutinizer. The Examiner process performs the following tasks: grade and update_marksheet. The Scrutinizer process performs the following tasks: verify_grade and verify_update_marksheet. The tasks of Examiner and Scrutinizer processes must be executed in the following order: grade → verify_grade → update_marksheet → verify_update_marksheet. In the skeleton code for Examiner and Scrutinizer given below, initialize and insert down and up operations of the given semaphores to ensure the above mentioned execution sequence.

semaphore e ← 1 semaphore s ← 0	// Initialize e and s

Process Examiner
{
 	
while(TRUE){
 	
Down e
grade();
 	
Up s
Down e
 	
update_marksheet():
 	
Up s
 	
}
 	
}
	Process Scrutinizer
	{
	
	while(TRUE){
	
	Down s
	verify_grade();
	
	Up e
	Down s
	
	verify_ update_marksheet():
	Up e
	
	}
	
	}

4. **[4 points]** Write one advantage and one disadvantage of an User-level Thread over a Kernel-level Thread.

Advantage: Context switching is cheaper.
Disadvantage: Less control of Kernel. A buggy user level thread may occupy CPU for a long time.