UECS2103/2403/2423 Operating Systems
Tutorial 4

1. What are the hardware approaches to enforce mutual exclusion?

2. What are the disadvantages in using machine instructions to enforce mutual exclusion?

3. Consider the following pseudo code below.

```
int x = 1, y = 2, z = 2;
```

```
   | P(){
P1|   y = y * z;
P2|   x = x + y + z;
P3| }
```

```
   | Q(){
Q1|   z = x / (y - z);
Q2|   x = x + y * z;
Q3| }
```

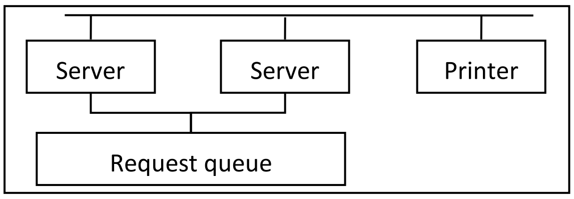
There are three global integers, x, y and z, considering the fact that any process can start first, what are the possible values of the variable **x**? For each value specified, state the situation or instructions execution sequence that yields to that value.
Assume that a process that generates error will be terminated and will not interfere another process. You may use the instruction number (P1, P2, etc.) in your explanation.

4. There are two processes sharing an I/O device.

   (a) Write the pseudo code that uses semaphore **s** to control the allocation of I/O device to the process. Assume that the function `write_data()` is called in the critical section.

   (b) Given that process P1 has higher priority than process P2, give a scenario where deadlock may occurs.

5. Consider two servers are sharing a request queue and a printer, as shown in the figure below. Each server will get a request from request queue, process it and send to printer. Printer and request queue can be accessed by only 1 server at a time. Write the pseudo code for the server process, use semaphores `p` and `q` to control the access to printer and request queue respectively. Assume that function `get_request()` gets request from the request queue and function `print()` will process the request and send it to the printer. Prevent the servers from accessing an empty queue. Initialise the values of semaphores appropriately.



6. Consider the pseudo code segments of two processes below.

```
     P(){
P1|    semWait(s);
P2|    m--;
P3|    semWait(t);
P4|    k = m * c;
P5|    semSignal(t);
P6|    semSignal(s);
     }
```

```
     Q(){
Q1|    semWait(t);
Q2|    m++;
Q3|    semWait(s);
Q4|    c++;
Q5|    semSignal(s);
Q6|    semSignal(t);
     }
```

Assume that both processes are assigned with the same priority level and process P executes first, is it possible that the process switching done by operating system can cause the two processes to enter a deadlock. Please explain. (You may use the line numbers as references in your explanation.)

7. A program consists of a few functions that perform different tasks as shown below. The `read_data()` function will read data from `data.dat` and store it in a shared buffer named `buf[]`. The `process_data()` function will get the data from `buf[]`, process it and update the outcome to `data.dat`. The variable `data_count` is used to store the number of data in `buf[]`, which will be reset to zero by process_data() after the data processing.

The program will first wait for request from user, once request is accepted, the `read_data()` function will be called. Once it is done, `process_data()` function will be called.

```
int buf[MAX_SIZE];
int data_count=0;

wait_for_request(){
  //wait and accept request from user
}

read_data(){
  //read data from data.dat into buf[]
}

process_data(){
  // get data from buf[]
  // process and update to data.dat
  // reset data_count to zero (0)
}
```

The program uses **binary semaphore** to control the access to buf[]. Rewrite the pseudo code for `read_data()` and `process_data()` with the use of semaphore to control the access of shared buffer. You must also prevent `process_data()` from reading an **empty** buffer and `read_data()` from writing to a **full** buffer. Use additional semaphores as needed.