CS 5348  Operating Systems, Homework #2

1. Consider the following program. Find a counter example (execution sequence) that demonstrates that this software solution is incorrect in terms of mutual exclusion violation.

   var blocked: array[0..1] of boolean;
   var turn: 0..1; // could be 0 or 1

   function p (id: integer)
   { repeat
       blocked[id] := true;
       while (turn ! = id) do
       { while blocked[1-id] do; // do nothing
           turn := id;
       }
       < critical section >
       blocked[id] := false;
       < remainder code for p>
       until false;
   }

   main ()
   { blocked[0] := false; blocked[1] := false;
     create thread to execute P(0);
     create thread to execute P(1);
   }

2. Consider a sequential program as follows. Assume that we try to execute all statements concurrently to achieve maximal parallelism, but we want to obtain the same output as the sequential execution below. Use (a) locks and (b) semaphores to control the synchronization so that the program can be executed in maximal concurrency while still get the same outcome as the sequential execution.

   x = y+z;
   w = y*z;
   u = w–3;
   v = x+t;
   k = x+w;

3. In the Jurassic Park Amusement center, tourists ride on a tour bus to visit the park. Each bus takes two visitors from the visitor center, drives around the park as long as the passengers wish, and return to the visitor center to drop off the passengers. There are N tour buses on service. If the N buses are all out, then a passenger who wants a ride must wait. If a bus is ready to load but there are no waiting visitors, then the bus waits. A bus does not leave until it loads two passengers. A bus does not load passengers if there is another bus that is only partly filled.

   Consider the problem of synchronizing the bus processes and the visitor processes using semaphores. The code for visitor process follows. Three semaphores cust, bus, and busReady are used. Write the code for the bus process that performs the proper operations on the three semaphores. Use extra semaphores as needed. In the bus process, you can use the same pseudo-statements "visitor get in
bus””, etc. as in the visitor process. Do not forget to properly initialize the semaphores (including the three given in the code and any new ones you may introduce).

```pascal
var cust: semaphore (:= ?);
    bus: semaphore (:= ?);
    busReady: semaphore (:= ?);

procedure visitor;
begin
    signal (cust);
    wait (bus);
    visitor gets in bus;
    wait (busReady);
    drive around the park;
    visitor gets off bus;
end;
```

4. Implement a monitor solution for the bakery problem. Assume that you have N salesmen and customers arrive at arbitrary times. The code for customers and salesmen external to the monitor is given as follows.

```pascal
customer
{ get_service ();
    receive the service
    release_service ();
}

salesman
{ prepare_service();
    provide the service
    complete_service();
}
```

5. When we implement the code for realizing the monitor, there is one important issue. Monitor can only allow one active thread in it. When signaling a condition variable, we need to decide whether to let the signaler or the signalee to continue. We have discussed how to implement the monitor with the choice of letting the signaler continue. Now, discuss the high-level implementation idea if we choose to let the signalee continue. You can give pseudo code and analysis like the “let signaler continue” case in the slide. (Hint: you may need to use another semaphore for conditional wait.)

6. Consider the bakery problem, but we replace the bakery by the hair salon. There are N hair dressers working for the salon. The customer can get the hair styling service if a hairdresser is available. The salon also has an entertainment center which has M seats. A customer can wait in the entertainment center for a hairdresser to become available. After finishing hair styling service, the customer makes payment to the salon.

A salon manager process coordinates the customers and hairdressers. It can admit at most N+M customers. A customer, when wanting to have the hair styling service, sends a message to the salon manager. The manager “receives” customers from the “custport” and place them in the entertainment center. If there is a waiting customer, the manager “receives” an available hairdresser from the “dresserport” and pairs the customer and the hairdresser for service. If there is no waiting customer,
the manager should not receive a hairdresser (hairdressers wait in a room). After the hair styling service finishes, the customer should make payment via the “payport”. The customer and hairdresser processes are given as follows. Give the code for the salon manager process using “Guarded Communication”.

Customer process:
   send (custport, customer);
   take a seat in the entertainment center and wait;
   get service from the hairdresser;
   send (payport, payment);

Hairdresser process:
   repeat select
      send (dresserport, hairdresser);
      provide hair styling service;
   until false;