T–79.4301 **Spring 2008**
Parallel and Distributed Systems
Tutorial 2 – Solutions

1. The algorithm does not correctly solve the mutual exclusion problem for two processes. For example, the following interleaving of statements executed by the two processes leads to an error:

Step	Process 0	Process 1
Initialisation: `want[0] := want[1]`		`:= false; turn := 0;`
1.		`i := mypid();` // i = 1
2.		`j := 1-mypid();` // j = 0
3.		`// [noncritical section]`
4.		`want[1] := true;`
5.		`(turn != 1)?` // true
6.		`(!want[0])?` // true
7.	`i := mypid();` // i = 0	
8.	`j := 1-mypid();` // j = 1	
9.	`// [noncritical section]`	
10.	`want[0] := true;`	
11.	`(turn != 0)?` // false	
12.	`// [critical section]`	
13.		`turn := 1;`
14.		`(turn != 1)?` // false
15.		`// [critical section]`

2. Promela model of Hyman's algorithm (with an assertion mechanism for checking its correctness):

```
/* Hyman's algorithm, two parallel processes 0 and 1 */

bool want[2] = false;  /* want[0..1] initially false */
bool turn = 0;         /* turn initialised to 0      */
byte count = 0;

active [2] proctype hyman()
{
    pid i = _pid, j = 1 - _pid;
    /* i is my index, j is the other process */

again:
    /* [noncritical section] */

    want[i] = true;
    /* [trying section] */
    do
    :: (turn != i) ->
        do
        :: (!want[j]) ->
            turn = i;
            break;
        :: else ->
            skip
        od
    :: else ->

        count++;
        assert(count == 1);
        /* [critical section] */
        count--;

        want[i] = false;
        break;
    od;
    goto again
}
```

Spin easily finds the model (`hyman.pml`) to be faulty:

```
$ spin -a hyman.pml
$ cc -o pan pan.c
$ ./pan
hint: this search is more efficient if pan.c is compiled -DSAFETY
pan: assertion violated (count==1) (at depth 51)
pan: wrote hyman.pml.trail

(Spin Version 5.1.3 -- 11 December 2007)
Warning: Search not completed
        + Partial Order Reduction

Full statespace search for:
        never claim             - (none specified)
        assertion violations    +
        acceptance   cycles     - (not selected)
        invalid end states      +

State-vector 20 byte, depth reached 51, errors: 1
       62 states, stored
       31 states, matched
       93 transitions (= stored+matched)
        0 atomic steps
hash conflicts:         0 (resolved)

    2.501       memory usage (Mbyte)


pan: elapsed time 0 seconds
```

3. The following execution leads to an error:

Step	Process 0	Process 1
Initialisation: `x := 0; y := 0; z := 0;`		
1.	`me := mypid()+1;` // me = 1	
2.	`// [noncritical section]`	
3.	`x := 1;`	
4.	`(y != 0 and y != me)?` // false	
5.	`z := 1;`	
6.	`(x != 1)?` // false	
7.		`me := mypid()+1;` // me = 2
8.		`// [noncritical section]`
9.		`x := 2;`
10.		`(y != 0 and y != me)?` // false
11.	`y := 1;`	
12.	`(z != 1)?` // false	
13.	`// [critical section]`	
14.		`z := 2;`
15.		`(x != 2)?` // false
16.		`y := 2;`
17.		`(z != 2)?` // false
18.		`// [critical section]`