HW 1

Teammates: Xiaocheng (xs2948)
Minke (

Q0

NAME	TACC USERID
Xiaocheng	Xiaocs1
Minke	

Q1

(a) When the intial State of Switch is off, assign one person as the counter. The counter is 0 at first. When the counter person enter the room, if the switch is off, he'll do nothing; Otherwise increment the counter by 1. For others, if the switch is on, do nothing; if it is the first time to find the switch is off, turn it on; Otherwise do nothing.

$$\text{Counter}\begin{cases}\text{on} \rightarrow \text{turn it off and counter ++}\\ \text{off} \rightarrow \text{continue}\end{cases}$$

$$\text{Others}\begin{cases}\text{on} \rightarrow \text{continue}\\ \text{off} \rightarrow \text{first time ? turn it on : Continue}\end{cases}$$

At the end, if the counter reaches P-1, then they can make an announcement to say all have entered the room.

(b) if the initial state of switch is unknown:

counter { on → turn it off and counter ++
counter { off → continue

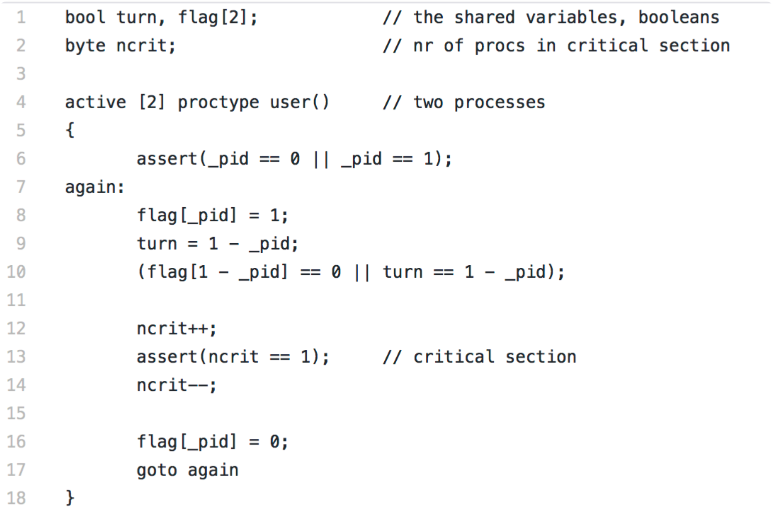
others { on → continue
others { off → first two times ? turn it on : continue

At the end, if the counter reaches 2 x (P-2) + 1 = 2P - 3, then they can make an announcement to say all have entered the room.

Q2

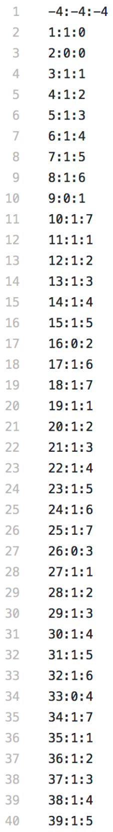
a)

Code:



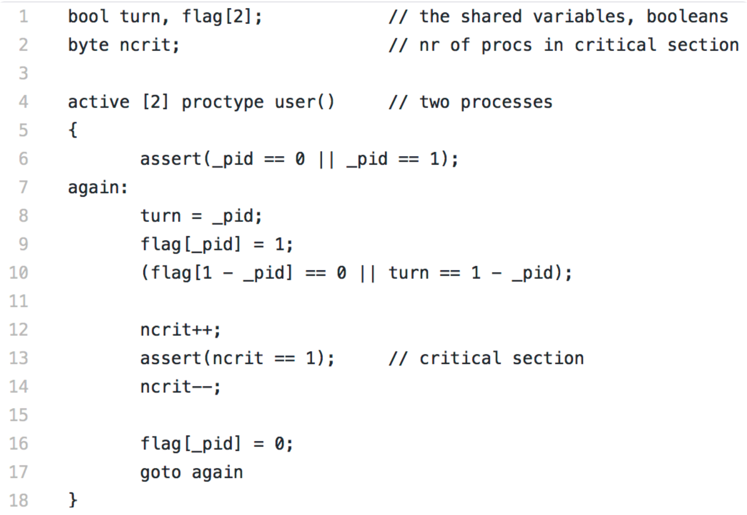
```
bool turn, flag[2];             // the shared variables, booleans
byte ncrit;                     // nr of procs in critical section

active [2] proctype user()      // two processes
{
        assert(_pid == 0 || _pid == 1);
again:
        flag[_pid] = 1;
        turn = 1 - _pid;
        (flag[1 - _pid] == 0 || turn == 1 - _pid);

        ncrit++;
        assert(ncrit == 1);     // critical section
        ncrit--;

        flag[_pid] = 0;
        goto again
}
```

Trail:



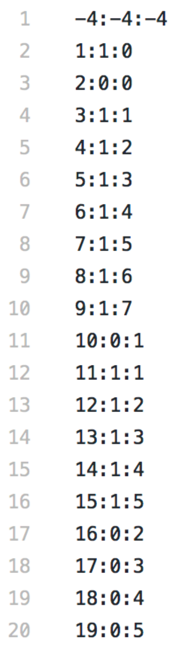
```
-4:-4:-4
1:1:0
2:0:0
3:1:1
4:1:2
5:1:3
6:1:4
7:1:5
8:1:6
9:0:1
10:1:7
11:1:1
12:1:2
13:1:3
14:1:4
15:1:5
16:0:2
17:1:6
18:1:7
19:1:1
20:1:2
21:1:3
22:1:4
23:1:5
24:1:6
25:1:7
26:0:3
27:1:1
28:1:2
29:1:3
30:1:4
31:1:5
32:1:6
33:0:4
34:1:7
35:1:1
36:1:2
37:1:3
38:1:4
39:1:5
```

b) Code:



```
bool turn, flag[2];             // the shared variables, booleans
byte ncrit;                     // nr of procs in critical section

active [2] proctype user()      // two processes
{
        assert(_pid == 0 || _pid == 1);
again:
        turn = _pid;
        flag[_pid] = 1;
        (flag[1 - _pid] == 0 || turn == 1 - _pid);

        ncrit++;
        assert(ncrit == 1);     // critical section
        ncrit--;

        flag[_pid] = 0;
        goto again
}
```

Trail:



```
-4:-4:-4
1:1:0
2:0:0
3:1:1
4:1:2
5:1:3
6:1:4
7:1:5
8:1:6
9:1:7
10:0:1
11:1:1
12:1:2
13:1:3
14:1:4
15:1:5
16:0:2
17:0:3
18:0:4
19:0:5
```

Q3

Suppose A has entered level 1, and is about to enter level 2. If A pauses for some reason, every other thread which completed level 1 may overtake A.

Q4

Assume there is N threads. So the previous gate level should be N-1, and now we reduce it as N-l, so that l threads can be in the CS. The modified code is as following:

```
for (int k = 1; k < N - l +1; k++) {
   gate[i] = k; //process i is in gate k
```

```
    last[k] = i; //process i is the last one to modify gate k
    int counter = l + 1;

    while (counter > l && last[k] == i) {
        counter = 0;
        for (int j = 0; j < N; j++) {
            if (gate[j] >= k) {
                counter++;
            }
        }
    }
}
```