HW #1

Mohit Joshi
msj696

3. msj696

4. a) a process sets "turn" to itself

Suppose that process 0 sets wantCS[0] = true and then sets turn = 0 instead of 1

Then process 1 sets wantCS[1] = true in another thread

Then process 0 evaluates (wantCS[1] && turn == 1) to false and enters its critical section

Then process 1 sets turn = 1 instead of 1, evaluates (wantCS[0] && turn == 0) to false and enters its critical section

Both processes are now in their critical sections at the same time, breaking mutual exclusion //

b) a process sets "turn" before setting "wantCS"

Suppose that process 0 sets turn = 1 instead of setting wantCS first

Then process 1 sets turn = 0, sets wantCS[1] = true, evaluates (wantCS[0] && turn == 0) to false, and enters its critical section

Then process 0 sets wantCS[0] = true, evaluates (wantCS[1] && turn == 1) to false, and enters its critical section

Both processes are now in their critical sections at the same time, breaking mutual exclusion //

5. Change int turn = 1 to boolean turn[2] = {false, true}

```
public void requestCS(int i) {
    int j = 1 - i;
    wantCS[i] = true;
    boolean temp = turn[j];
    turn[i] = temp;
    while (wantCS[j] && temp == turn[j]);
}
```

6. Consider the scenario where some processes p_0 and p_1 start choosing numbers concurrently while all other numbers are still 0

Then p_1 reaches the number[i] ++ line when p_0 finishes its second iteration of the previous line

In this scenario, their numbers will be the same when they do the check to enter their critical sections

Before p_0 increments its number, p_1 does the check and enters its critical section, then p_0 increments, checks, and enters its own critical section

Both processes are in their critical sections at the same time, breaking mutual exclusion. //