


```
int total;
int grandTotal = -1;

int myRoutine(int a, int b) {
    int product, left, right;
    product = a * b;
    left = a;
    right = b;
    total = 0;

    while (left > 0) {
        total += right;
        left--;
    }

    grandTotal = grandTotal + (total - product);

    return grandTotal;
}
```

- a) Is this code safe? If so, argue informally but convincingly that it is. If not, modify it to make it thread safe. (You may use synchronization primitives, rearrange code, etc)

3. (hard) The goal of this exercise is for you to create a monitor with methods Hydrogen() and Oxygen(), which wait until a water molecule can be formed and then return. For example, if two threads call Hydrogen() and then a third calls Oxygen(), the third thread should wake up the first two threads and then they should all return.

a) Specify the correctness constraints. Be succinct and explicit.

Observe that there is only one condition any thread will wait for (that a water molecule can be formed). However, it will be necessary to signal hydrogen and oxygen threads independently. Consequently, we use two condition variables, waitingH and waitingO.

Define wH and wO to be the number of hydrogen and oxygen threads waiting in the monitor, and define aH and aO as the number of assigned threads waiting in the monitor. All variables are initialized to 0. Start with the following code:

```
Hydrogen() {
    wH++;
    lock.acquire();
    while (aH == 0) {
        if (wH >= 2 AND wO >= 1) {
            wH -= 2; aH += 2; wO -= 1; aO += 1;
            waitingH.broadcast();
            waitingO.signal();
        }
        else {
            lock.release();
            waitingH.wait();
            lock.acquire();
        }
    }
    lock.release();
    aH--;
}
```

```
Oxygen() {
    wO++;
    while (aO == 0) {
        if (wH >= 2 AND wO >= 1) {
            wH -= 2; aH += 2; wO -= 1; aO += 1;
            waitingH.signal();
            waitingH.signal();
        }
    }
}
```

```
        }  
        else {  
            waitingO.broadcast();  
        }  
    }  
    aO--;  
}
```

b) Does Hydrogen() work, not work, or is it dangerous (works intermittently)? If it is broken, explain how to fix it.

c) How about Oxygen? If it is broken, explain how to fix it.

4. (Medium) A simple implementation of a lock is given below. Why must the interrupts be disabled in the **Release** procedure?

```
class Lock {
    int value = FREE;

    Acquire() {
        Disable interrupts
        if (value == BUSY) {
            put on wait Q
            sleep
            disable interrupts
        } else {
            value = BUSY;
        }
        Enable interrupts;
    }

    Release() {
        Disable interrupts;
        if (wait queue not empty) {
            take thread off wait Q
            add to ready Q
        } else {
            value = FREE;
        }
        Enable interrupts;
    }
}
```

