Board examples - critical sections, race conditions

ex

```
int balance = 50;

1  void withdraw (int amount) {
2      if (balance > amount) {
3          cout << "approved";
4          balance -= amount;
5      }
   }
```

2 threads start

T1 withdraw 40 } 40+25 = 65 > 50 !
T2 withdraw 25 }

| sequence | Balance After |
|---|---|
| T1 starts gets to 3 (past the filter) | 50 |
| T1 preempted by T2 | 50 |
| T2 runs from 1 → 5 | 25 |
| T1 switched back in | -15 |

race condition - when program output depends on what gets done first.

# race condition

ex.

```
int global = 2;

void fun() {
    if (global == 0)
        doZero();
    else
        doNotZero();
}

int main() {
    thread myt(fun);
    global = 0;
    myt.join();
}
```

execute doZero() or doNotZero()?

if ① happens before ② doZero();

if ② happens before ① doNotZero();

how can you tell?

you cannot as written

you can however use condition variables to synchronize these (later)

PSA you may see code that "fixes" this with delays (sleep_for(), sleep_until()) this is a cheesy non scalable solution. Do not do this

critical section - an area of code where only 1 thread can be at a time.

Question: if you launch no threads, do you have CS's?

Answer: No, but design your app to be ported from single to multiple threads. That is, Identify, mark, condense critical sections comment them but dont add synchronization code (kills performence!)

ex.

```
int gi=0;

void fun(){
    gi++;
}

int main(){
// thread t1(fun)
    int i=gi;
    i = i+1;
    gi = i;

t1.join();
```

where are critical section(s)

if no thread

if fun just reads gi

if thread started in ① ② or ③

① start here → // thread t1(fun)

② or → int i=gi;

③ or → i = i+1;

gi = i;