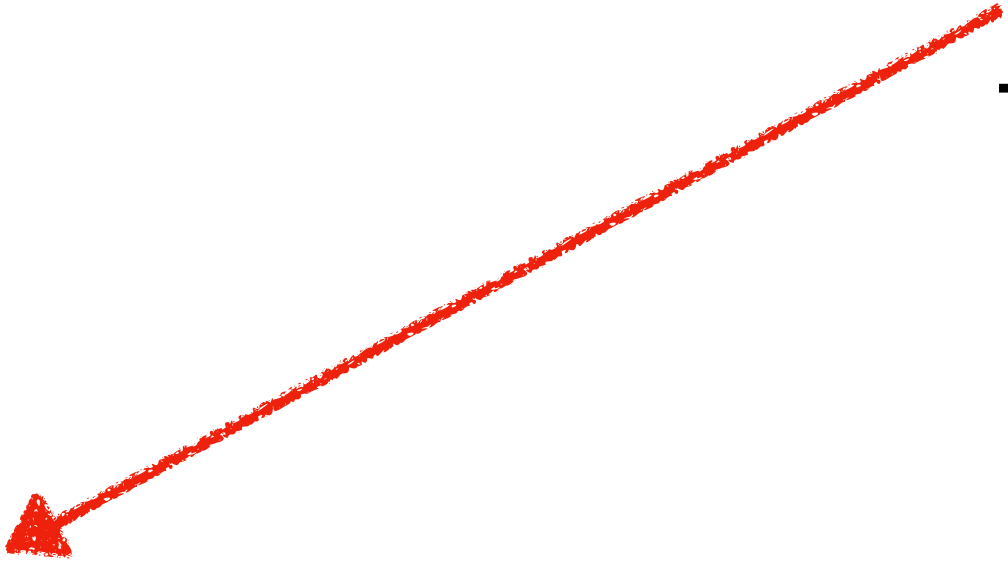
41

```
bool threadRunning;

bool proveFermatsLastTheorem() // Thread 1 {
    threadRunning = true;
    for (int n = 3; threadRunning; ++n) {
        if (pow (x, n) + pow (y, n) == pow (z, n)) {
            return false;
        }
    }
    return true;
}

void testTheorem () {
    bool result;
    startThread ([] () (result = proveFermatsLastTheorem));
    Sleep (2000);
    threadRunning = false;
    std::cout << result << std::endl;
}
```

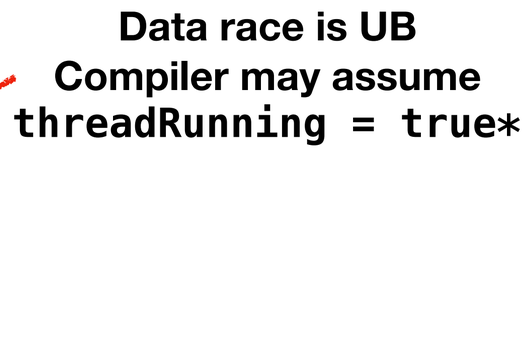


Data race is UB

Compiler may assume

`threadRunning = true`*

*** A valid C++ compiler is allowed to assume `threadRunning` is always `true` (due to data-race being UB). Most compilers we tested will nevertheless do the right thing here (i.e. check `threadRunning` every iteration) - but they are not required to. Your code may break when you update compiler versions for example.**




```
bool threadRunning;

bool proveFermatsLastTheorem() // Thread 1 {
    threadRunning = true;
    for (int n = 3; threadRunning; ++n) {
        if (pow (x, n) + pow (y, n) == pow (z, n)) {
            return false;
        }
    }

    return true;
}

void testTheorem () {
    bool result;
    startThread ([] () (result = proveFermatsLastTheorem));
    Sleep (2000);
    threadRunning = false;
    std::cout << result << std::endl;
}
```

*** A valid C++ compiler is allowed to assume `threadRunning` is always `true` (due to data-race being UB). Most compilers we tested will nevertheless do the right thing here (i.e. check `threadRunning` every iteration) - but they are not required to. Your code may break when you update compiler versions for example.**

```
bool threadRunning;

bool proveFermatsLastTheorem() // Thread 1 {
    threadRunning = true;
    while (true) {
        if (pow (x, n) + pow (y, n) == pow (z, n)) {
            return false;
        }
        ++n
    }

    return true;
}

void testTheorem () {
    bool result;
    startThread ([] () (result = proveFermatsLastTheorem));
    Sleep (2000);
    threadRunning = false;
    std::cout << result << std::endl;
}
```