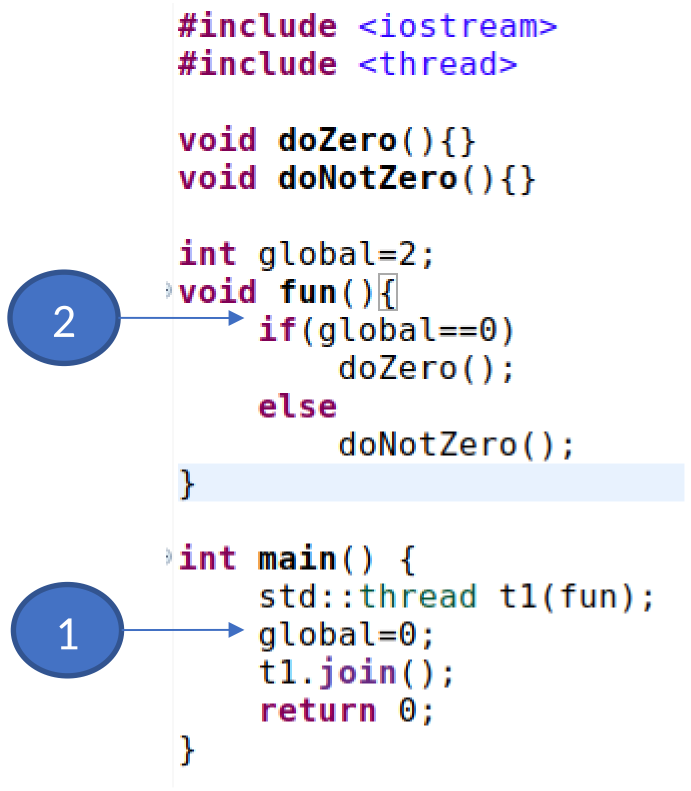
Race Condition again- what happens here?

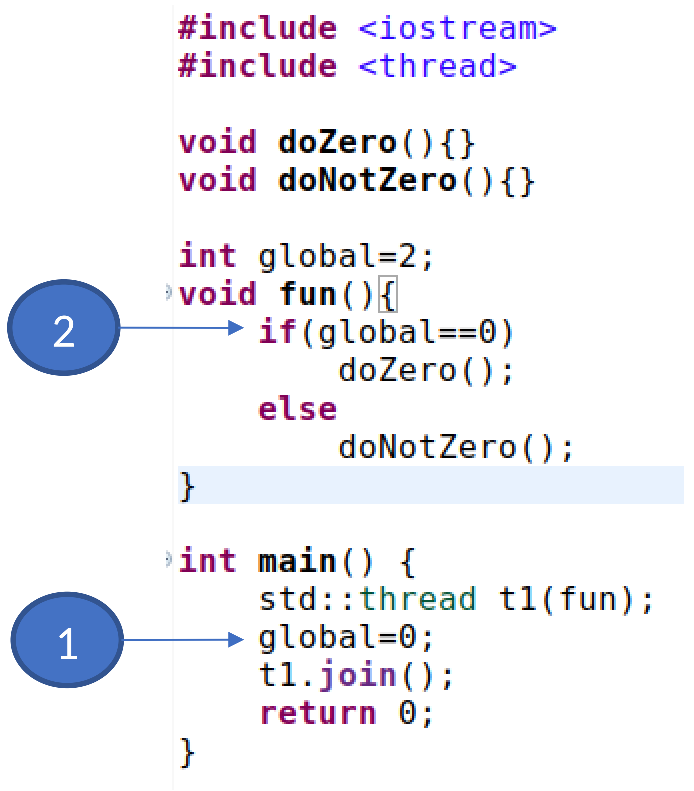


```
#include <iostream>
#include <thread>

void doZero(){}
void doNotZero(){}

int global=2;
void fun(){
    if(global==0)
        doZero();
    else
        doNotZero();
}

int main() {
    std::thread t1(fun);
    global=0;
    t1.join();
    return 0;
}
```

- Do you execute doZero() or doNotZero()?
- If 1 happens before 2
  - Then doZero()
- If 2 happens before 1
  - Then doNotZero()

How can you tell what happens?

Race Condition again- what happens here?



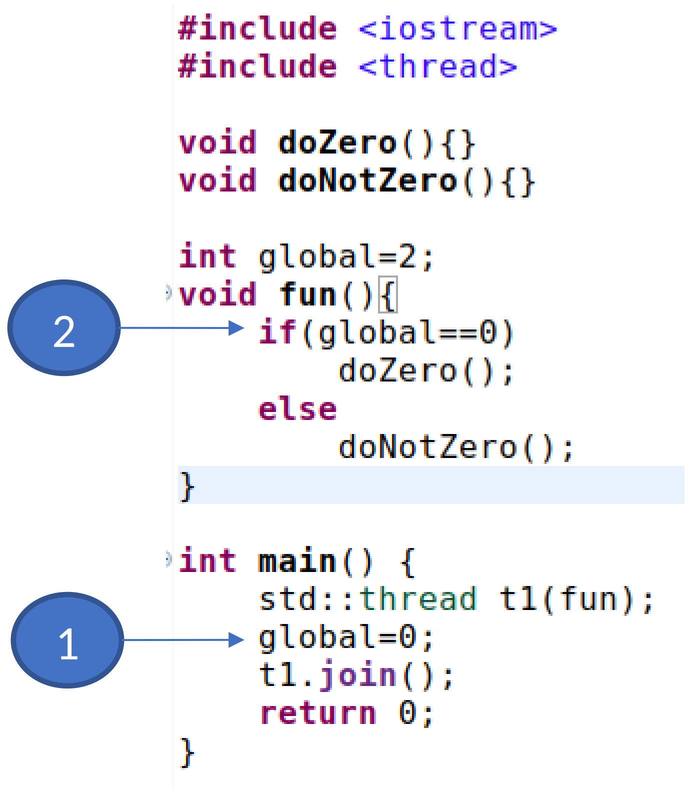
```
#include <iostream>
#include <thread>

void doZero(){}
void doNotZero(){}

int global=2;
void fun(){
    if(global==0)
        doZero();
    else
        doNotZero();
}

int main() {
    std::thread t1(fun);
    global=0;
    t1.join();
    return 0;
}
```

- Do you execute doZero() or doNotZero()?
- If 1 happens before 2
  - Then doZero()
- If 2 happens before 1
  - Then doNotZero()

How can you tell what happens?

You cannot as written.

Race Condition again- what happens here?



```
#include <iostream>
#include <thread>

void doZero(){}
void doNotZero(){}

int global=2;
void fun(){
    if(global==0)
        doZero();
    else
        doNotZero();
}

int main() {
    std::thread t1(fun);
    global=0;
    t1.join();
    return 0;
}
```

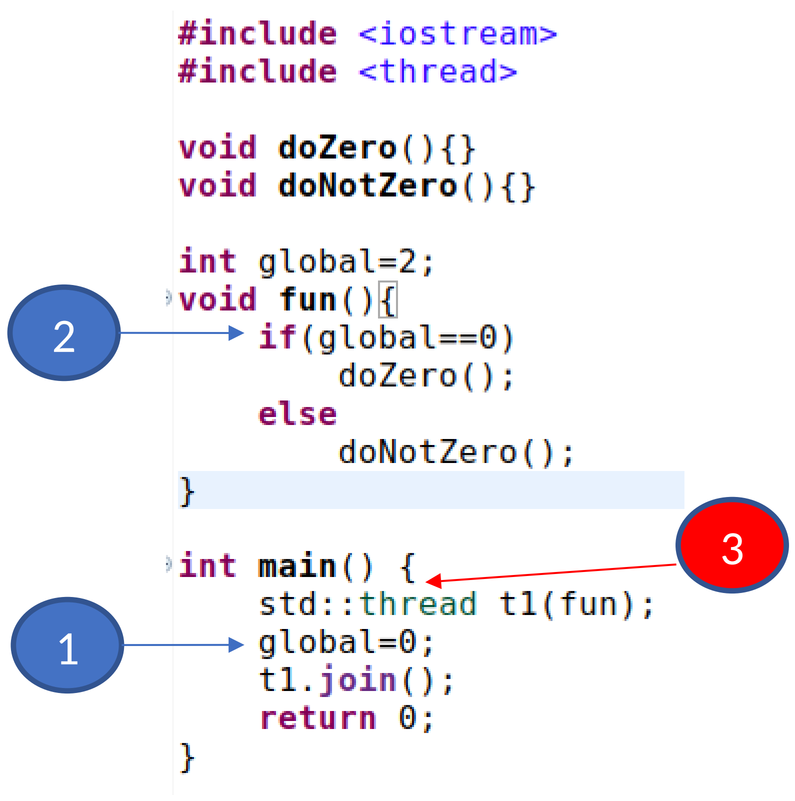
- Do you execute doZero() or doNotZero()?
- If 1 happens before 2
  - Then doZero()
- If 2 happens before 1
  - Then doNotZero()

How can you tell what happens?

You cannot as written.

You can however use condition variables to impose an order of your choice (later)

Race Condition again- what happens here?



```
#include <iostream>
#include <thread>

void doZero(){}
void doNotZero(){}

int global=2;
void fun(){
    if(global==0)
        doZero();
    else
        doNotZero();
}

int main() {
    std::thread t1(fun);
    global=0;
    t1.join();
    return 0;
}
```

- Do you execute doZero() or doNotZero()?
- If 1 happens before 2
    - Then doZero()
- If 2 happens before 1
    - Then doNotZero()

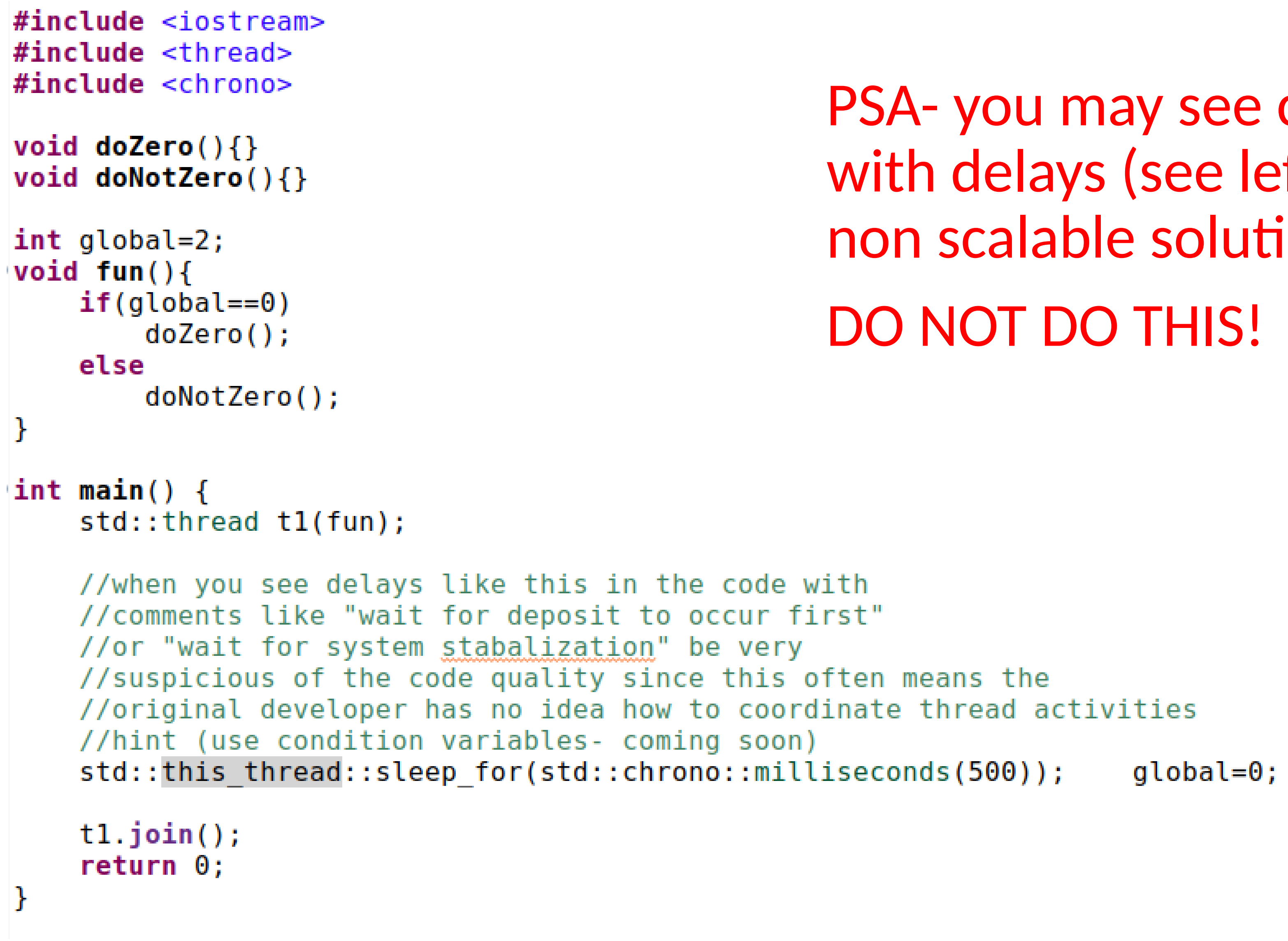
How can you tell what happens?

You cannot as written.

You can however use condition variables to impose an order of your choice (later)

Or move 1 to position 3

Race Condition again- A bogus solution



```cpp
#include <iostream>
#include <thread>
#include <chrono>

void doZero(){}
void doNotZero(){}

int global=2;
void fun(){
    if(global==0)
        doZero();
    else
        doNotZero();
}

int main() {
    std::thread t1(fun);

    //when you see delays like this in the code with
    //comments like "wait for deposit to occur first"
    //or "wait for system stabalization" be very
    //suspicious of the code quality since this often means the
    //original developer has no idea how to coordinate thread activities
    //hint (use condition variables- coming soon)
    std::this_thread::sleep_for(std::chrono::milliseconds(500));    global=0;

    t1.join();
    return 0;
}
```

PSA- you may see code that “fixes” this with delays (see left). This is a cheesy, non scalable solution. (Why?)

DO NOT DO THIS!