Where are critical sections in the following code?

```
int g=0;

void fun(){
    g++;
}

int main(){
    thread t1(fun);

    int i=g;

    i++;

    g=i;

    t1.join();:
}
```

1

If no threads?
If no threads then single threaded, no critical sections.

If fun() just reads g?
g is being written at
If 1 write then all reads and writes need protection
See code in the rounded rectangle for critical sections

If thread starts in position 1

Where are critical sections in the following code?

```
int g=0;

void fun(){
    g++;
}

int main(){
    thread t1(fun);

    int i=g;

    i++;

    g=i;

    t1.join();:
}
```

1

If no threads?
If no threads then single threaded, no critical sections.

If fun() just reads g?
g is being written at If 1 write then all reads and writes need protection
See code in the rounded rectangle for critical sections

If thread starts in position 1

Where are critical sections in the following code?

```
int g=0;

void fun(){
    g++;
}

int main(){
    int i=g;

    thread t1(fun);

    i++;

    g=i;

    t1.join();:
}
```

2

If no threads?
If no threads then single threaded, no critical sections.

If fun() just reads g?
g is being written at
need protection

If 1 write then all reads and writes

See code in the rounded rectangle for critical sections

If thread starts in position 2

Where are critical sections in the following code?

```
int g=0;

void fun(){
    g++;
}

int main(){
    int i=g;

    thread t1(fun);

    i++;

    g=i;

    t1.join();:
}
```

2

If no threads?
If no threads then single threaded, no critical sections.

If fun() just reads g?
g is being written at
need protection
See code in the rounded rectangle for critical sections

If 1 write then all reads and writes

If thread starts in position 2

Where are critical sections in the following code?

```
int g=0;

void fun(){
    g++;
}

int main(){
    int i=g;

    i++;

    g=i;

3   thread t1(fun);

    t1.join();:
}
```

If no threads?
If no threads then single threaded, no critical sections.

If fun() just reads g?
g is being written at If 1 write then all reads and writes need protection
See code in the rounded rectangle for critical sections

If thread starts in position 3

Where are critical sections in the following code?

```
int g=0;

void fun(){
    g++;
}

int main(){
    int i=g;

    i++;

    g=i;

3 →  thread t1(fun);

    t1.join();:
}
```

If no threads?
If no threads then single threaded, no critical sections.

If fun() just reads g?
g is being written at ( ) If 1 write then all reads and writes need protection
See code in the rounded rectangle for critical sections

If thread starts in position 3

As written, only t1 will access g when the application
Is multithreaded. So there are no critical sections for position 3
as the global is never accessed in a multithreaded environment.

Where are critical sections in the following code?

```
int g=0;

void fun(){
    g++;
}

int main(){
    int i=g;

    i++;

    g=i;

3 →  thread t1(fun);
    g--;

    t1.join();:
}
```

If no threads?
If no threads then single threaded, no critical sections.

If fun() just reads g?
g is being written at ( ) If 1 write then all reads and writes need protection
See code in the rounded rectangle for critical sections

If thread starts in position 3

As written, only t1 will access g when the application
Is multithreaded. So there are no critical sections for position 3 as the global is never accessed in a multithreaded environment.

But it only takes a slight change to the code to cause problems!
These types of changes often occur over the lifetime of the codebase