101

```
int main () safe
```

}

```
println (*s);
```

```
auto shared_data = shared_ptr<mutex<string>>::make(string ("Hello threads"));
```

```
for(int i : num_threads)
```

{

```
string^s = lock_guard^.borrow();
```

```
s^->append ("🔥");
```

```
void entry_point (shared_ptr<mutex<string>> data, int thread_id) safe
```

```
vector<thread> threads { };
```

```
auto lock _guard = data->lock();
```

```
threads^. push_back(thread (&entry_point, copy shared_data, i));
```

```
const int num threads = 15;
```

```
apply ([tid] (auto& s) {
```

```
void entry_point (std::shared_ptr<synchronized_value<std::string>> sync_s, int tid)
```

```
s.append("🔥");
```

```
return s;
```

},

```
auto s = std::make_shared<synchronized_value<std::string>> ("Hello threads");
```

```
*sync_s);
```

{

```
std::vector<safe_thread> threads { };
```

}

```
const int num_threads = 15;
```

```
int main()
```

```
std::println ("{} {}", s, tid);
```

```
threads.push_back (safe_thread (entry_point, auto (s), auto (i)));
```

```
for (int i : std::views::iota (0, num_threads))
```

```
copy shared_data, i));
```

```
shared_ptr<mutex<string>> data,
```

{

```
int main() safe
int thread_id) safe
```

```
s^->append("🔥");
```

}

//...

```
threads^.push_back(thread (&entry_point,
void entry_point (
auto lock_guard = data->lock();
```

```
println (*s);
```

```
string^s = lock_guard^.borrow();
```

```
void entry_point (
s.append("🔥");
```

```
int tid)
```

```
apply ([tid] (auto& s) {
```

```
std::shared_ptr<synchronized_value<std::string>> data,
```

{

```
std::println ("{} {}", s, tid);
```

```
*data);
```

```
int main()
```

```
return s;
```

```
auto (s), auto (i)));
```

```
threads.push_back (safe_thread (entry_point,
```

}

```
},
```

//...