- Exercise

Let's solve an exercise about locks.

WE'LL COVER THE FOLLOWING

- Problem statement

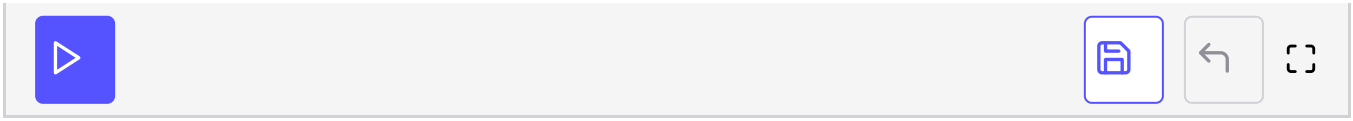
Problem statement

We should not explicitly use a mutex.

- Adjust a lock to the given program.
- Which lock is suitable? (`std::unique_lock` or `std::lock_guard`)

```
#include <chrono>
#include <iostream>
#include <mutex>
#include <string>
#include <thread>

std::mutex coutMutex;

class Worker{
public:
  Worker(std::string n):name(n){};

    // Update the worker class to achieve the functionality

    void operator() (){
      for (int i = 1; i <= 3; ++i){
            // begin work
            std::this_thread::sleep_for(std::chrono::milliseconds(200));
            // end work
            //coutMutex.lock();
            std::cout << name << ": " << "Work " << i << " done !!!" << std::endl;
            //coutMutex.unlock();
      }
        }
private:
  std::string name;
};


int main(){
```

```
int main(){

  std::cout << std::endl;

  std::cout << "Boss: Let's start working." << "\n\n";

  std::thread herb = std::thread(Worker("Herb"));
  std::thread andrei = std::thread(Worker("  Andrei"));
  std::thread scott = std::thread(Worker("    Scott"));
  std::thread bjarne = std::thread(Worker("      Bjarne"));
  std::thread andrew = std::thread(Worker("        Andrew"));
  std::thread david = std::thread(Worker("          David"));

  herb.join();
  andrei.join();
  scott.join();
  bjarne.join();
  andrew.join();
  david.join();

  std::cout << "\n" << "Boss: Let's go home." << std::endl;

  std::cout << std::endl;

}
```



In the next lesson, we'll discuss the solution to this exercise.