- Solution

Let's look at the solution to the exercise.

WE'LL COVER THE FOLLOWING

Solution

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

std::mutex coutMutex;

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

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


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"));
```

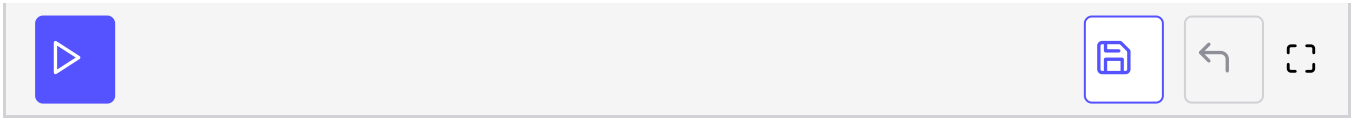
```
  std::thread bjarne = std::thread(Worker("      Bjarne"));
  std::thread andrew = std::thread(Worker("        Andrew"));
  std::thread david = std::thread(Worker("          David"));


  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;

}
```



Explanation

The boss assigns three work packages (lines 13 - 21) to each of its six workers (lines 33 - 38). When a worker is done with its work package, it screams out loudly to the boss (line 19). When the boss has been notified by all workers, it sends them home (line 47).

By invoking the method `std::lock_guard<std::mutex> myCoutLock(coutMutex);`, we define the exclusive section. This section can only be accessed by, at most, a single thread. The access to `std::cout` is synchronized and the mess becomes harmony.

In the next lesson, we'll discuss the thread-safe initialization of variables in concurrent programming with C++.