1. Mutex

```cpp
#include <semaphore>

class Mutex
{
public:
  Mutex() = default;
  Mutex(Mutex const &) = delete;
  Mutex & operator=(Mutex const &) = delete;

  void lock()
  {
    semaphore_.acquire();
  }

  void unlock()
  {
    semaphore_.release();
  }


private:
  std::binary_semaphore semaphore_ {1};
};
```

Поля: semaphore – бинарный семафор.

Методы: lock() – вызывает у семафора acquire(), который в свою очередь атомарно декрементирует внутренний счетчик на 1 если он больше 0, иначе же блокирует поток. Unlock() – вызывает у семафора release(), который в свою очередь атомарно инкрементирует внутренний счетчик на 1. В случае если есть заблокированные потоки которые ждали, что внутренний счетчик станет больше 0, то они разблокируются.

2. Вывод

При замене std::mutex на Mutex будет следующий вывод:

```
=== CookQuit ===
gluttony1: 50, gluttony2: 50, gluttony3: 50 efficiency_factor 100
dish1: 5500, dish2: 5500, dish3: 5500
Eaten:
fat_man1: 2500, fat_man2: 2500, fat_man3: 2500
CookQuit

=== CookNoSalary ===
gluttony1: 1000, gluttony2: 1000, gluttony3: 1000 efficiency_factor 1000
dish1: 3000, dish2: 3000, dish3: 3000
Eaten:
fat_man1: 11000, fat_man2: 11000, fat_man3: 11000
CookNoSalary

=== CookIsFired ===
gluttony1: 50, gluttony2: 50, gluttony3: 5000 efficiency_factor 1
dish1: 3001, dish2: 3001, dish3: -1999
Eaten:
fat_man1: 0, fat_man2: 0, fat_man3: 3001
CookFired
```