Mutex Lock Operations

```
shared bool available = true
```

```
void acquire () {
   while (!available)
      ; // do nothing
   available = false;
}
```

```
void release (){
   available = true;
}
```

Classical Semaphore Operations

```
void wait (semaphore &s)
{
   while (s <= 0)
      ; // do nothing
   s--;
}
```

```
void signal (semaphore &s)
{
   s++;
}
```

Blocking Semaphore Operations

```
struct Semaphore {
   int value;
   list<pid_t> l;
};
```

```
void wait(Semaphore &s) {
   s.value--;
   if (s.value < 0) {
      s.l.push_back(PID);
      block(PID);
   }
}
```

```
void signal (Semaphore &s) {
   s.value++;
   if (s.value <= 0) {
      PID = s.l.front();
      s.l.pop_front();
      wake(PID);
   }
}
```