Mutex:  
- **lock**: Acquire the lock, blocking if necessary.
- **unlock**: Release the lock. Assumed that the calling thread owns the lock.

C++ Mutex:  
```cpp
mutex m;
unique_lock<std::mutex> lk(m);
lk.lock();
lk.unlock();
```

Condition Variables:  
- **wait**: Block the calling thread until the condition has been signaled. Atomically does the following:
  1. Releases lock.
  2. Add thread to the waiters for cond.
  3. Sleeps thread until awoken.
- **signal**: Signal that the condition has been met, awakening a single waiting thread. (Though not switching to the newly awoken thread immediately.)
- **broadcast**: Signal that the condition has been met, awakening all waiting threads.

C++ Condition Variables:  
```cpp
condition_variable cv;
cv.wait(unique_lock<std::mutex> lock);
cv.notify_one();
cv.notify_all();
```

Multi-Threaded BFS Pseudocode:
```cpp
while (1) {
    lk.lock()
    while (queue.empty() & !done)
        wait(lk)

    // Have lock
    if (done)
        lk.unlock()
        break

    data = queue.remove()
    lk.unlock()

    // process data
    lk.lock();
    q.add(new_data);
    lk.unlock();
    signal();
}
```