Deadlock Solutions

Deadlock

- Explain what is meant by deadlock
  - A thread is deadlocked when it cannot run
- Give an example of how deadlock can occur
  - Thread A acquires a lock on mutex 1 and waits for a lock on mutex 2
  - Thread B acquires a lock on mutex 2 and waits for a lock on mutex 1
- Suggest two ways to avoid deadlock
  - Always acquire locks in the same order
  - Use language features which can acquire multiple locks in a single operation

Deadlock

- Write a program which causes two threads to deadlock
- Implement your solutions. Verify that the program is no longer affected by deadlock

Deadlock avoidance (contd)

- In the following code, why are the unique_lock objects needed when the mutexes are already locked?

```
lock(mutex1, mutex2);
unique_lock<mutex> lk1(mutex1, std::adopt_lock);
unique_lock<mutex> lk2(mutex2, std::adopt_lock);
```

  - To ensure that the mutexes are always unlocked when leaving the enclosing scope
- Rewrite the solution to use unique_lock objects with the defer_lock option