CPSC 427 Final Week

Build base app: DUE 12/6 I will give you an API to build from. You must use these to build your project. I will also give you a test cpp file to put your database through its paces.

Thread Review

Context switch -> unload current, load next and run

Do student demo in class to show why needed
Context switch when I say, stop remember where you are then start

```
Void threadFunc(std::string name){
        for (int i=0;i<10;i++){
                cout<< 'I am'+ name;
        }
}
```

Goes in this order
Person1->person2->person3->person1
Draw command window
Start
Make sure you stop some in the middle of writing
Show errors

Fix:

Critical section is cout<< 'I am'+ name;
Add a lock around it;

```
Mutex myMutex;      //global var
```

One will aquire by locking, the others wait

Give them a Pink Mutex paper
Do round robin
Stop before done writing, do context switch
Others waiting to lock() bummer waste their time waiting
Fix (use try_lock() and yield())
Get around to original person
Finishes locks again when unlock switch

DEADLOCK

```
void threadFuncKeith(std::string name){
        myMutex1.lock();
        myMutex2.lock();
        //process here
}

void threadFuncStudent(std::string name){
```

```
        myMutex2.lock();
        myMutex1.lock();
        //process here
}
```

Threadfunckeith aquires myMutex1 context switches

threadFuncStudent aquires myMutex2 waits on myMutex1
context switches
Threadfunckeith waits on myMutex2

Frozen, deadlock

Fix always aquire in same order.

Asserts

See 21_Asserts project
F12 go to assert definition
Note the conditional compilation
In preprocessor a void function is inserted, optimizing compiler sees that knows its nothing so puts in nothing. Presto conditional compilation

Casting

Prefer C++ style casts

Review Friday;

Pointers – see quiz
Classes
Heiarchies (may ask you to design something)
- **Virtual**
- **Abstract**
- **Base**
- **Derived**
- **Overridden**

Composition verses Inheritance when to use which
Threads