CS 6/73201 # Project #2 **Advanced OS**

Due via email by 11:59pm on Friday 21 April 2000

The Problem

In this assignment, you are to (i) use Nachos to implement **atomic transactions and concurrency control for a database** (as described in Lectures 18 and 19), and (ii) come up with appropriate **test code or methods** to demonstrate that your implementation works for at least 3 copies of Nachos accessing a datatbase maintained by yet a 4th copy of nachos. Take part (ii) seriously — think about how you can have these copies of Nachos run, make requests to the database using transactions, contend for access in such a way that the concurrency control is needed, etc., and then make sure that your implementation and test cases demonstrate this effectively. This project will comprise 20% of your final course grade.

What to Turn In

(30 points) In a file called **p2.overview**, write a brief overview of your project. Describe what you implemented, any design decisions that you made, and why you made those decisions. Describe your test cases, and how they demonstrate that the project work. Since the TA will have a large number of different projects to grade, supply instructions to the TA as to how he should test your project to demonstrate that it works. Do a good job on this writeup — convince the TA and me that you implemented something interesting, that you learned something from doing it, and that you actually had some tough choices to make in your implementation. A superficial one-page writeup will definitely not be worth 30 points!

(70 points) The implementation of your project and test code / methods. If it doesn't quite work as desired, clearly describe in the file **p2.overview** what you have done, what is not working, and how you would go about finishing the project if you had more time. The better you describe the status, the better your chances will be for getting partial credit for what you've done. Also, if you have something that works, and you can document that status with test cases, etc., you'll probably get more credit than if you have a lot of code written, but nothing that compiles (since in the latter situation it's harder for the TA to evaluate the status of your work).

Identifying Your Changes

So that the TA and I can easily identify which code you have changed or added, surround all changes and additions in your code by comments in the following form:

```
// PROJECT 2 CHANGES START HERE
<your changed code goes here>
// PROJECT 2 CHANGES END HERE
```

Use your own judgment about how much code to surround in a single comment.

Where to Get Help

Same as Project 1

Cooperation versus Cheating

Same as Project 1

Submitting Your Project

When you finish, submit <u>all files</u> that you modified to the TA, just as you did in Project 1.

The project is due at 11:59pm on Friday 21 April 2000. For a discussion of my late policy, see the class syllabus. However, you should probably plan on starting early, ending on time, and then spending the weekend resting or working on something else, instead of trying to perfect a late project.