Midterm Examination

CS 169

Oct. 29, 2002

Please read all instructions, including these, carefully. This is a closed-book, open-note exam (you may have four sheets of notes). There are 5 questions on the exam, all in multiple parts. All questions have a short, concise answer. Complex or unnecessarily long solutions may receive as little as no credit; clarity in your answers is as important as giving correct answers. Please be sure to clearly mark your answers. Partial answers will be graded for partial credit.

Name: Sam Pell Solution

SID: _____

Problem	Points	Score
1	20	
2	20	
3		

20

4
20

5
20

1. Dynamic Analysis (20 points) In class, we discussed the Eraser algorithm for detecting memory synchronization errors. Consider the following code. Assume a single thread executes each statement in the order given, and that the if statement on line 16 evaluates to true. Fill in the contents of other columns in the table to their values after the statement on the corresponding line is executed. Some rows are already filled in for you.

Line	Code	C(x)	C(y)	locks held
		{a,b,c}	{a,b,c}	{}
1	lock(a)	{a,b,c}	{a,b,c}	{a}
2	lock(c)	{a,b,c}	{a,b,c}	{a,c}
3	x=x+1	{a,c}	{a,b,c}	{a,c}
4	unlock(c)	{a,c}	{a,b,c}	{a}
5	y=y+1	{a,c}	{a}	{}
6	unlock(a)	{a,c}	{a}	{}
	...			
15	lock(b)	{a,c}	{a}	{b}
16	if (y>0) {	{a,c}	{}	{b}
17	lock(c)	{a,c}	{}	{b,c}
18	x=x+1	{c}	{}	{b,c}
19	unlock(c)	{c}	{}	{b}
20	}	{c}	{}	{b}
21	unlock(b)	{c}	{}	{}

Which lock(s) protect the variable x? C

Which lock(s) protect the variable y? none

Would Eraser produce a warning? Yes, on line 16

Is there a bug in the given code? If so, what? Yes, data race on y on line 16

There was some confusion on this problem concerning whether the Eraser algorithm in question was the basic algorithm or the one that distinguished read locks from write locks. We meant for you to only consider the basic algorithm; since you don't know from the problem statement if the locks held are held in read or write mode, it doesn't really make sense to solve the problem assuming otherwise, but if you gave it a shot anyway, we tried to grade in as generous a manner as possible.

Now consider the following code. A program may or may not need the value of

pi to 1000 digits. Since this is an expensive computation, it will only

compute the value if needed, and at most only compute it once, using the

following Java code. Note that the method compute_pi may be called many times by different threads, but the body of the method is supposed to compute the value of pi only one time.

```
1    protected Object _my_lock = new Object ();
2    protected BigDecimal _cached_value;
3    public BigDecimal compute_pi () {
4        if (_cached_value == null) {
5            synchronized (_my_lock) {
6                if (_cached_value == null) {
7                    // do actual computation, putting result in
8                    // the variable _cached_value
9                }
10           }
11       }
12       return _cached_value;
13   }
```

Would Eraser issue a warning about this code, and if so because of what line?

Eraser would issue a warning on line 4.

Will the code compute pi more than once? Why or why not?

No. It will only compute pi once. Even though more than one thread may reach line 5, the check inside the synchronized block will ensure that only one thread performs the calculation.

This problem is taken almost verbatim from the paper, p. 404 and translated into Java from C. The motivation for the two checks (lines 4 and 6) is explained there.

Many people got hung up on the Virgin/Exclusive/Shared/Shared-Modified state machine. We didn't really mean for you to worry about that, but if you did, the answer is as follows: the variable _cached_value is initialized to null as the class is loaded into the JVM, putting it in the Exclusive state. Thread 1 executes up to and including line 4, then takes a timer interrupt. Thread 2 executes up to and including line 4, and also takes a timer interrupt, placing the variable in the Shared state. Thread 1 wakes back up, acquires the lock, and writes the variable, placing it in the Shared-Modified state. Sometime later, Thread 3 comes around and executes line 4, an access to the variable unprotected by a lock. Hence a warning is issued.

2. Architecture (20 points) In a pipeline architecture data passes through a series of stages. A diagram for a typical pipeline architecture is shown below; the boxes are stages of the pipeline and the edges represent flow of data from one stage to the next. In each case a box on the left performs its computation and then ships data to the box on its right.

One problem with a pipeline architecture is what to do about errors. Assume each data value may cause any stage to raise an error. What modifications would you make to the pipeline architecture to deal gracefully with errors? Feel free to draw a modified diagram, but please also provide some text explaining your solution. Do not assume that you can simply add a new first stage that filters out all data that might cause an error in any subsequent stage.

The simplest solution is to have each stage shunt all invalid inputs to a separate error-handling module that generates appropriate error messages and discards the bad data. The architecture would look like:

Common mistakes: Some answers assumed a particular kind of pipeline; for example, that we were talking about a hardware pipeline. A hardware pipeline is a pipeline, but there are lots of uses of the pipeline software architecture other than in hardware. The purpose of an architecture question is to focus on the properties that pertain to all examples of that architecture. Some answers assumed that data could be "cleaned" and fed back into the pipeline, or that the whole system should abort on the first error. Again, these might be necessary or possible for particular pipeline systems, but are certainly not

characteristic of all pipelines. Also, some people just made this question much more complicated than necessary.

Another common architecture superficially resembles a pipeline, but the flow of control is completely different because one intermediate stage X must aggregate all of the data and maintain it in some way that requires a fairly expensive computation (e.g., sorting it). X does not push data to subsequent stages automatically, but sends it only on demand, in response to requests. In the diagram below, boxes are stages, data always flows from left to right, and the direction of arrows shows which side initiates the transfer of data. What is a real-world example of such an architecture?

Dealing with updates to the data collected in stage X requires thought. Assume that the leftmost stage sends stage X new data on the order of once per hour, that this data is to be added to the data that X has already collected, that the amount of new data is always small relative to what X already has, and that it takes X one day to recompute its function of the data. The recomputation requires one day regardless of how much new data was received, and during this time X cannot respond to requests. Give a strategy for managing updates to X. Can you come up with an improved architecture where X is always available to answer requests, and if so, what?

X

A relational database or a web search engine is an example of such an architecture.

Given that the time to recompute is insensitive to the number of updates, it makes sense to batch updates together, say on the order of once per day or once every other day. To improve availability, we could duplicate X. Version 1 of X will be serving at any time while version 2 is recomputing. When version 2 is done recomputing, they switch roles: version 1 is now recomputing while version 2 is serving. In this way we could always be responsive to requests and the maximum delay between an update arriving and being in available for queries would be less than 48 hours. Some web search engines are structured in exactly this way.

Common mistakes: Many students forgot to answer the first part of the question.
3. (Testing and Checking, 20 points) The Traveling Salesman Problem is defined as follows: you are given an undirected graph where the nodes are cities and the edges have positive weights which are distances between cities. Given a list of cities to visit, find an order that minimizes total distance traveled.

Solving the traveling salesman problem optimally is believed to require time exponential in the number of cities, so practical implementations often use quite complex heuristics. Assume that you are given an implementation of Traveling Salesman which you may treat as a black box (i.e., you may give it problems to solve and look at the answers, but you cannot do anything else). How would you automatically test such an implementation? Please give three different and practical automatic tests that would improve your confidence in the implementation of the algorithm. For each test, please explain why it will be effective.

There are many possible tests; here is a representative selection.

Enumerate small examples, say with 5-8 cities, find the optimal solution by brute force (by enumerating all possible tours) and compare with the solution found by the implementation. Since there are n! tours for n cities, this test does not scale beyond more than a few cities but will be very effective at finding small examples that don't work, which are likely to point out problems with the system.

On larger examples with a tour of N cities, try removing each city and compare the results for each of these N-1 tests on N-1 cities with the N city result. The N-1 city tests should all return tours that are less costly than the N city test. This greatly multiplies the number of test cases we get from every test case entered by humans and gives confidence that the system is at least monotone---that adding cities makes tours more expensive.

Construct a path with a known least cost tour. This graph can be "padded" with lots of extra cities and expensive edges that would not be included in the minimal cost tour. In this way we can create a huge example with a known minimal tour. Testing the system on such examples will give information about scalability and sensitivity of the answers to problem size.

Common mistakes: Many solutions suggested strategies that required examining the code or doing a second implementation. Both are out of bounds, as the problem says specifically all you can do is run the program. Other solutions gave very generic answers that didn't provide any insight into what to do for this particular problem. For example, answers would say "use random testing", without explaining how to generate a random TSP instance, which is actually not at all obvious.

Some answers claimed TSP is an NP-complete problem, and thus verification of the output is P-time, and that we could exploit this to write a fast checker. But, only the problem TSP-k is NP-complete: "Given a map of cities and distances, is there a tour through all the cities of length less than k." Here we do not know k as part of the input, and so there is no obvious fast checker.
This question has nothing to do with the first half of problem 3. Consider a sorting implementation that correctly sorts a sequence of numbers with probability .9. That is, on certain input sequences the implementation always returns an incorrect answer, but these "bad" sequences are relatively rare and distributed randomly throughout the set of all input sequences. How would you construct a version that sorts correctly with probability at least .999?

Prepend a random integer to the head of the list. This gives us a randomly chosen list (at least randomly chosen among all lists with one new element at the head, which is good enough) that we can sort. Repeat until there are at least three different lists that are the same, where we check for sameness by comparing the resulting lists element-wise, skipping over the randomly inserted integer in each list. The probability that all the lists will be the same (modulo the extra integer) and incorrect is .1 * .1 *.1, or <= .001.

Common Mistakes:

Many answers assumed that the sorting algorithm was non-deterministic, even though the problem statement clearly says that it always returns wrong answers on certain inputs.

Consequently, many students proposed running the sort on the same sequence 3 times in a row to get higher accuracy.

Many answers did not include a step to check the validity of the sort after each run. For example, some students proposed taking the initial sequence, sorting it, then sending the modified sequence through the sort (without checking to see whether the modified sequence dropped any elements).

Some answers proposed inefficient solutions that required running the sort 1000 times or more and then taking the most popular sequence.

Finally, some students gave solutions that involved debugging the sorting algorithm itself (via correctors, asserts, and random testing).

4. (Runtime Analysis & Memory Management, 20 points) In lecture we discussed three different applications for runtime analysis of software (software fault isolation, detecting memory corruption, and detecting data races). Recall that a memory leak in a garbage-collected language happens when the garbage collector cannot reclaim storage because the data is still reachable but the program is in fact no longer using it. Propose a runtime analysis technique for detecting such memory leaks in Java. You should clearly state your algorithm, discuss its memory and time overhead, and explain any compromises you made or limitations that the algorithm may have.

With each region of memory, v, allocated by the program, allocate two shadow words: one, PC(v), stores the program counter value when the variable was allocated; the other, r(v), is initially zero. (A region could be as fine grain as every word of memory, or as coarse as every object.) Every time the program accesses the region we assign r(v):=0. During garbage collection, if v is not collected then we increment r(v). If r(v)>threashold for any v, we issue a warning of a possible leak, printing out v, r(v), and PC(v). A debugger can then be used to find the line of code where the variable was allocated.

The memory overhead is at most 3X, which is completely reasonable for this type of tool. (Eraser, for example, uses at least that much.) The cost in speed comes from three places. First, with every read or write to an address, v, we must clear r(v). If we're smart, we

only allocate v in the range 0xc0000000 and above. Then we set r(v)=v & 0x7fffffff and PC(v)=v & 0x3fffffff. So in assembly, for example, a read of v

```
load r1, v
```

becomes

```
move r1, v
set  r1, r1 & 0x7fffffff
stor r1, 0
load r1, v
```

We get a slowdown of 4, assuming all instructions take the same time, and we don't use any additional registers.

This was a hard problem. If you used something similar to the above, you got full credit. One simple change that still works is to store the time of last access in r(v) instead of a counter.

Common mistakes: The most common problem was failure to read the directions, which specifically say that a memory leak "happens when the garbage collector cannot reclaim storage because the data is still reachable". Many answers to this problem boiled down to trying to improve the garbage collector or use reference counting, but since the data is still reachable, none of these help. The next most common problem with the answers given was that many people wanted to free suspected leaked memory. You can't do that. If you knew it was a leak, and the program could really be shown to never use it again, you could free it, and you'd have a great new garbage collector. Otherwise, freeing suspected leaks may introduce bugs. Finally, a lot of people proposed some sort of code analysis to find memory that the program couldn't use again. This is very difficult, and besides, the problem asked for a runtime analysis.

5. (Miscellaneous Short Answer, 20 points)

What is mutation analysis and what is its purpose?

Mutation analysis deals with changing code in the program to create “mutant” programs. These mutant programs are used to test the test suite’s adequacy.

Common Mistakes:

Some students thought mutation analysis dealt with testing the adequacy and accuracy of the code itself.

You are trying to apply delta debugging to a search program that takes binary trees as input and returns a node in the tree or reports failure. You have a test case that fails stored in a large file. The tree is stored in preorder, so the first node in the file is the left-most descendant of the root, and the last node in the file is the root of the entire tree. To minimize the test case, you repeatedly split the file in half, performing binary search. Why is this likely to be an extremely poor use of delta debugging? What would be a better way to apply delta debugging?

Cutting the file in half probably will not produce valid input---half the file is unlikely to be a tree at all. A much better way of using delta debugging is to parse the file into a tree data structure, and then subdivide the input down by taking individual subtrees.

Common Mistakes:
Students thought that ordering (preorder/inorder/postorder) mattered.

A common design for GUI’s is to have every gesture correspond to a statement in a small language which is actually executed by an interpreter separate from the GUI itself. What is the advantage of this use of the Command pattern?

The command pattern helps to automate GUI testing. It also helps create a layer of abstraction between the GUI and application logic layer (therefore it is easier to port an application to another system).

Common Mistakes:

Some students said that it's easier to add more commands. This is not really true, as adding a new command requires adding code in two places (the gesture in the UI, and the new command in the interpreter) instead of one.

In lecture we discussed the difference between a file-oriented and a project-oriented version control system. The difficulty with a file-oriented system is that it may be difficult or impossible to recover an arbitrary past state of the project. Succinctly explain why file-oriented systems have this problem, and how project-oriented systems solve it.

File-based systems do not tie modification times to a project version. So it's not easy to find out what file versions are associated with each release. Project-oriented systems take snapshots of the versions of all files at a given time thus allowing for easy access to older builds.

Common Mistakes:

None