

CS249 Final Exam: Closed Book, 180 Minutes

Instructor: David Cheriton, Monday December 9, 2004

December 3, 2007

This is a CLOSED-BOOK exam. You are expected to work on your own to complete this exam, not using other people or any materials for your assistance.

- Please answer all questions; read each question carefully. You waste time by writing more than required by the question, and you lose points by answering the wrong question.
- Please give precise, specific answers that demonstrate the depth of your knowledge of the area being examined by the question, rather than vague and general comments. Point form answers are acceptable. You can use sketch code that is not necessarily complete AS LONG AS IT CLEARLY INDICATES you understand the approach or technique we are after.
- Please put your name and student number on each exam booklet you use and sign the honor code statement.

1. 30 Points

- (a) Assuming you are using the attribute-only interface approach, describe how inline read accessors and data members in the interface class compromise the separating of interface from implementation, or argue that this approach does not (and justify your answer).
- (b) If you can use default values and range-constrained index types to eliminate exceptions from read accessors, why can't you use the same approach with write accessors, or can you? Be specific!
- (c) Junior programmer (smart) Alex claims he accepts everything in CS249 except the attribute-only interface approach and wants to use verbs in class interfaces. Describe three key issues that you would expect him to encounter, i.e. other aspects of cs249 that depend on the attribute-only interface approach.

2. 30 Points Consider a String class implemented as:

```
class StringBuffer : public PtrInterface<StringBuffer> {
    . . .
};

class String {
    . . .
private:
    StringBuffer::Ptr strBuf_;
};
```

That is, `String` simply consists of a (smart) pointer to a string buffer containing the actual string data.

- (a) Describe the category of the `String` type - i.e. value, entity or shared description, and justify why it should be so considered.

- (b) Given that `String` just consists of the single data member `strBuf_`, you might think it would be simpler, more efficient and less error-prone to just use `StrBuffer` directly. Sketch some code you would expect to see in a full implementation of `String` sufficient to argue that using `StrBuffer` directly would confer none of these benefits.
- (c) Consider providing a `GeometricDesc` class that indicates an object's its position and orientation in 3-D space as well as its geometry. Sketch the code for this class at the same level of detail as given for `String`, indicating how position, orientation and geometry are implemented, justifying your design.
3. **30 Points** Considering events and notification:
- (a) describe as specifically as possible the three key advantages of the `Activity` approach over using conventional event scheduling or threads.
- (b) describe how to apply the call-up/notify-down approach to the structure of software in a network switch (or another embedded application that you are familiar with) and describe how the separation into configuration and reporting interfaces impacts this design.
4. **30 Points** In C++, it seems necessary, at first, to enclose every function call in a a try block, and even each catch statement in a try block yet Cheriton claims it is feasible to write many functions with no try blocks. In fact, he claims that in many cases, you only need to consider adding a try block later as an optimization.
- (a) List the three key requirements for this approach and how each is achieved in the course material.
- (b) Sketch a brief C++ function that illustrates adding a try block as an optimization. You can provide commentary describing how it fits into the overall system or program and why this is "just an optimization".
5. **30 Points** Cheriton makes the claim that the `Ptr` class template is basically the only form of smart pointer required, pushing other functionality to proxy objects.
- (a) Illustrate three key advantages of this approach as compared to the alternatives with code snippets for each advantage (and associated commentary).
- (b) Cheriton also claims that `Ptr` allows you to engineer *predictability* into the performance and resource requirements of your software, compared to automatic garbage collection. Describe what this means, describe why this is hard to achieve with garbage collection, and describe a case to be careful of even with `Ptr` to achieve this.
6. **30 points** Considering composition and collisions:
- (a) One approach to providing a framework timeout facility is to define an interface class
- ```
class TimeableObj {
 void virtual timeout() = 0;
};
```
- and require that application objects derive from this. Describe the alternative approach advocated in the course and illustrate three key advantages of this alternative with a specific example.
- (b) Consider the situation in a flight simulation with an airplane and a missile, and the airplane takes off and then in the course of flight fires the missile. Describe three key points about the interaction between the airplane and the missile and the rest of the virtual world relating to the composition and collision topics covered in the course. (There are many right answers — impress us with insight!)

The End, Happy Holidays!