Midterm Exam
CS169, Fall 2005
November 9, 2005

- Please read all instructions (including these) carefully.
- Write your name, login, and SID.
- There are 8 pages in this exam and 8 questions, each with multiple parts. Some questions span multiple pages. All questions have some easy parts and some hard parts. If you get stuck on a question move on and come back to it later.
- You have 1 hour and 20 minutes to work on the exam.
- No electronic devices allowed, including cell phones used as watches.
- The exam is closed book, but you may refer to your two pages of handwritten notes.
- Please write your answers in the space provided on the exam, and clearly mark your solutions. You may use the backs of the exam pages as scratch paper. Please do not use any additional scratch paper.
- Solutions will be graded on correctness and ***clarity***. Each problem has a relatively simple and straightforward solution. Partial solutions will be graded for partial credit.

LOGIN: _____

NAME: _____

SID: _____

Problem	Max points	Points
1	6	
2	5	
3	5	
4	12	
5	18	
6	18	
7	18	
8	18	
TOTAL	**100**	

Problem 1: True/False [6 points, 1 each]

For each question, answer true or false. You can explain your view if needed below.

(a) Refactoring is typically performed to fix bugs.

(b) Automated tools can detect bugs in the source code but not in the formal design of the system.

(c) Prototypes help in gathering requirements.

(d) Regression tests are developed early in the XP process.

(e) Sandboxing relies on virtual memory hardware.

(f) Use Case Diagrams are useful in specifications but not in designs.

A	
B	
C	
D	
E	
F	

Problem 2: Processes [5 points]

(a) [3] Formulate a short example of a Customer Test, as used in XP Programming.

(b) [2] Give an example of a "world change" that may necessitate a change in the design. Describe the change in the form of a (changed) user requirement.

Problem 3: Design Patterns [5 points]

(a) [2] Which pattern commonly extends the Composite pattern?

(b) [3] Give the motivation for the Bridge pattern.

Problem 4: Delta Debugging [12 points]

Question 1 [6 points]

Given the following (initial) sequence of tests applied by a delta debugging algorithm, mark all minimal subsets of changes that could be returned by the algorithm.

A	B			?
		C	D	?
A				?
	B			✓
		C		?
			D	✓
	B	C	D	?
A		C	D	✗

{A}	
{C}	
{D}	
{A,B}	
{A,C}	
{A,D}	
{B,C}	
{C,D}	
{A,C,D}	

Question 2 [6 points]

After the following initial sequence of tests, one can infer that A must be part of any minimal subset returned by the algorithm. Explain why this is so.

A				✓
	B	C	D	✓
A	B		D	✗

Problem 5: Testing [18 points]

Question 1 [8 points]

In certain scenarios, testing can be *complete*, in that it is guaranteed to find all bugs in the tested code. Describe conditions that must hold for testing to be complete.

Give an example of a program on which testing can be complete.

Question 2 [10 points]

This question asks you to describe how you would test a method that computes an *intersection* of two *multisets*.

> **Background:** *Multisets* are extensions of sets that allow multiple instances of the same element. For example, {1,2,2} is not a set but it is a multiset. *Intersection* is computed as with sets, except that the resulting multiset can contain multiple instances of an element *e*: the quantity of *e* is the minimum of *e*'s quantities in the two multisets.

When creating test cases, you usually want the test cases to "cover" certain *properties*. For example, in testing a division method, you may want have at least one test cases that divides by zero.

Your goal is to list up to three such properties for testing the multiset intersection procedure.

1)

2)

3)

Problem 6: UML [18 points]

Question 1 [8 points]

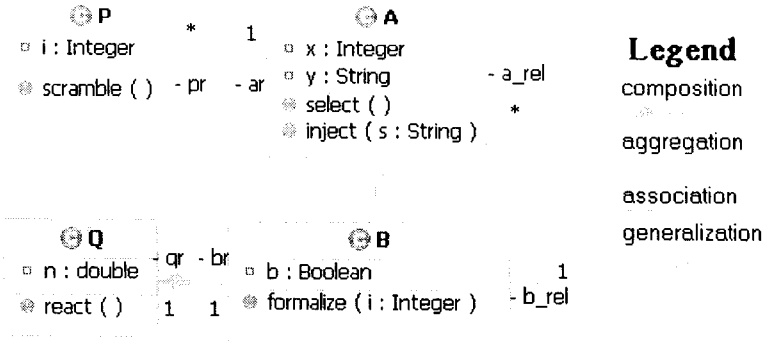
Match the UML terms in the left column with *the most appropriate* description in the right column. Write your answers in the space provided.

a. aggregation	1. transitive containment relationship (does not allow cycles)
b. association	2. exclusive containment relationship (no one else can contain the same object)
c. generalization	3. ability of one object to send a message to another
d. composition	4. ability of one object to access another's private data
	5. relationship between two objects with complementary functionality
	6. relationship between two objects where one extends the functionality of another

a ___ b ___ c ___ d ___

Question 2 [10 points]

Generate skeleton Java code from the following (nonsensical) UML diagram (see next page). The answer will contain four public Java classes (A,B,P,Q), their fields and their methods (leave the bodies of methods empty). Your code must incorporate the relationship names that label the arrows. It should also be possible to compile your code. If you use Java container classes (e.g. List, Set) you should write comments indicating the type of the contained objects; alternatively you may use arrays or Java 5 generics.


P
i : Integer
scramble ( )
*
1
- pr
- ar
A
x : Integer
y : String
select ( )
inject (s : String)
- a_rel
*
Q
n : double
react ( )
- qr
- br
1
1
B
b : Boolean
formalize (i : Integer)
1
- b_rel
Legend
composition
aggregation
association
generalization

Problem 7: Concurrency Bugs [18 points]

Question 1 [6 points]

Define datarace. Three conditions must hold for a datarace to occur:

1)

2)

3)

Question 2 [6 points]

Mark all dataraces in the code below. Note that to mark a datarace, it is not sufficient to identify a single statement.

Thread 1	Thread 2

```
if(f == null)
        lock(L)
        if(f == null) {
                f = fopen("foo");
        }
        unlock(L);
}
```

```
lock(L);
if(f == null) {
        f = fopen("foo");
}
unlock(L);
```

Question 3 [6 points]

The following code contains a multithreading bug. Write some code (performed by some other thread) that manifests the bug.

```
lock(buff);
len = buff.length ();
unlock(buff);
...
...
lock(buff);
c = buff.getChar(len - 1);
unlock(buff);
```

Describe in what way is the bug in manifested: _____

_____

Problem 8: Project [18 points]

Question 1 [7 points]

Argue for or against:

"Having a formal specification document helps in implementing a nontrivial system".

You must base your argument on examples from your CS169 project. In particular, explain whether you should have spent more or less time on gathering and formalizing specifications for your (non-trivial) project.

Question 2 [11 points]

Describe the unit testing strategy for one of the components in your project. You need not go into great detail, so please keep it short.

- Write a brief description of the component:

- Write an outline of the unit testing strategy:

- Give two unit tests that verify *different aspects* of the component. For each unit test describe the input, the processing to be performed on that input, and the expected results.