UNIVERSITY OF EDINBURGH

COLLEGE OF SCIENCE AND ENGINEERING

SCHOOL OF INFORMATICS

INFR11101 ADVANCES IN PROGRAMMING LANGUAGES

Tuesday 12 th May 2015

14:30 to 16:30

INSTRUCTIONS TO CANDIDATES

Answer any TWO questions.

All questions carry equal weight.

CALCULATORS MAY NOT BE USED IN THIS EXAMINATION

Year 4 Courses

Convener: I. Stark
External Examiners: A. Cohn, T. Field

THIS EXAMINATION WILL BE MARKED ANONYMOUSLY

1. This question is about concurrent programming with threads.

   The following Java is from a class written to represent a person's name. Some of the code is to be used in concurrent programming.

```
// Class to represent a person's name, made up of their first name and last name.
public class FullName {

    private String first = "";
    private String last = "";

    // Operation to copy the contents of one fullname into another
    public static void copy(FullName p, FullName q) {
        q.first = p.first;
        q.last = p.last;
    }

    // Wrapper around the copy operation
    public static void safe_copy(FullName p, FullName q) {
        synchronized(p){             // Claim first fullname p
            synchronized(q){         // Claim second fullname q
                copy(p,q);           // Copy across contents
            }
        }
    }

    // Remainder of class omitted
}
```

   (a) Describe what it means for methods in Java to be *thread safe*. [*1 mark*]

   (b) The copy method is not thread safe. Explain why, showing fragments of code and their execution to demonstrate how this can be a problem in practice. [*6 marks*]

   (c) The method safe_copy is a wrapper around copy that is intended to be thread safe. However, it is still problematic for use in concurrent code, as it may cause *deadlock*.

      (i) Describe what it means for threaded concurrent code to *deadlock*. [*2 marks*]

      (ii) Explain why safe_copy may cause deadlock, and give code fragments demonstrating how this can be a problem in practice. [*6 marks*]

QUESTION CONTINUES ON NEXT PAGE

(d) Many concurrent systems use *thread priority* to manage scheduling of thread execution. High-priority threads are allocated processor time over lower-priority threads; perhaps even interrupting their execution.

The *Mars Pathfinder* lander, described in lectures, used a priority scheduling system. A few days after landing on Mars in July 1997 the lander software ran into deadlock problems caused by *priority inversion*. Every time the system deadlocked, automatic safety monitors reset the software to restart afresh the next day; and every day it deadlocked again.

(i) Describe briefly what is *priority inversion* and how it may cause deadlock. [*4 marks*]

(ii) The Pathfinder system had the following components.

- A memory bus shared among many processes, with access controlled by a lock.
- A low-priority weather observation process L that occasionally posted data to the bus.
- A medium-priority long-running communications task M that did not use the bus.
- A high-priority process H to regularly check that all was well on the bus.

Summarize the sequence of events involving these components that can lead to a priority inversion deadlock. [*6 marks*]

2. This question is about types and type systems.

(a) The following are three variations on the idea of *polymorphism* in programming languages.

(i) Subtype polymorphism.

(ii) Parametric polymorphism.

(iii) Ad-hoc polymorphism.

For each of these give a brief explanation of what it is, and give an example. [*6 marks*]

Note: Each example can be in any programming language — Haskell, Java, Scala, or whatever you think appropriate — but you must say which language it is. You can use different languages for each example if you think that will help your explanations.

(b) Suppose we have a dependently-typed lambda calculus which includes types *Int* of integers, *Num* of non-negative integers, and $\mathit{Matrix}\,n\,m$ of integer matrices with n rows and m columns, for $n, m : \mathit{Num}$. One possible operation in the language is to generate an identity matrix:

$$\mathit{identity} : \forall n{:}\mathit{Num}.\mathit{Matrix}\,n\,n\ .$$

(i) Give a suitable dependent type for the operation of matrix addition *add*.

(ii) Give a suitable dependent type for matrix multiplication *mult*.

(iii) Use some or all of *identity*, *add*, and *mult* to write out a term that computes the 5×5 matrix that has leading diagonal elements all 2 and zero elsewhere (i.e. double the identity matrix). [*5 marks*]

(c) System F extends the simply-typed lambda-calculus with explicit polymorphism: terms that take a type as a parameter. This language is expressive enough to define conventional algebraic datatypes from scratch. For example, if we assume predeclared types *Int* of integers and *Bool* of booleans, then we can define a type *Prod* of pairs of these.

$$\mathit{Prod} \stackrel{\mathit{def}}{=} \forall X.(\mathit{Int} \to \mathit{Bool} \to X) \to X$$

Give definitions of the following three terms: to extract the first and second components of such a pair, and to build a pair given *Int* and *Bool* arguments.

$$\mathit{fst} : \mathit{Prod} \to \mathit{Int}$$
$$\mathit{snd} : \mathit{Prod} \to \mathit{Bool}$$
$$\mathit{pair} : \mathit{Int} \to \mathit{Bool} \to \mathit{Prod}$$

Terms should be written with Church-style typing, giving explicit types for all arguments in each lambda-abstraction. [*6 marks*]

QUESTION CONTINUES ON NEXT PAGE

(d) Suppose that we now have a dependently-typed lambda calculus with types and terms to support a *deep embedding* of propositional logic. These include type *Prop* of propositions, type *ProofOf*(p) for proofs of p for each $p : \mathit{Prop}$, and the following terms.

$$\begin{aligned} \mathit{and} &: \mathit{Prop} \to \mathit{Prop} \to \mathit{Prop} \\ \mathit{proj1} &: \forall p, q{:}\mathit{Prop}.\ \mathit{ProofOf}(\mathit{and}\ p\ q\ ) \to \mathit{ProofOf}(p) \\ \mathit{proj2} &: \forall p, q{:}\mathit{Prop}.\ \mathit{ProofOf}(\mathit{and}\ p\ q\ ) \to \mathit{ProofOf}(q) \\ \mathit{conj} &: \forall p, q{:}\mathit{Prop}.\ \mathit{ProofOf}(p) \to \mathit{ProofOf}(q) \to \mathit{ProofOf}(\mathit{and}\ p\ q\ ) \end{aligned}$$

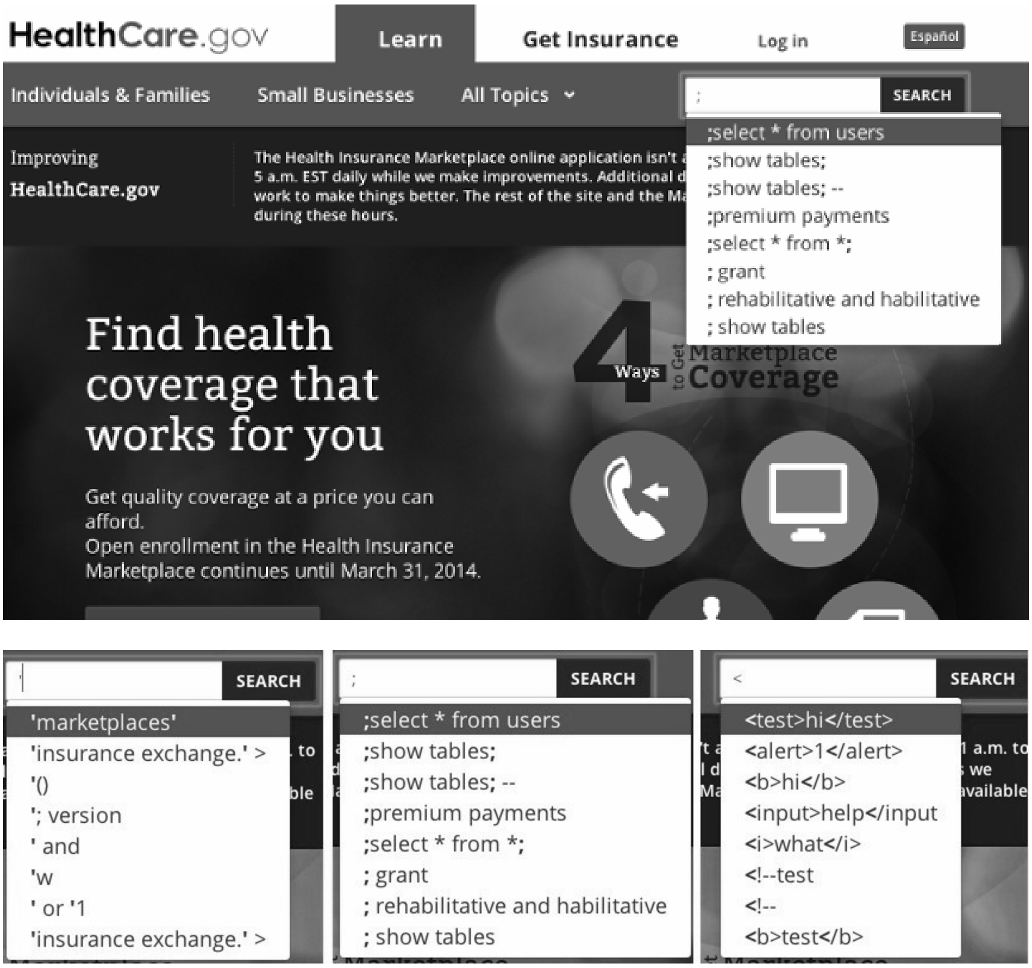
(i) What logical proof principle is captured by the term *proj1*?

(ii) Use these terms to give a term with the following type:

$$\forall p, q{:}\mathit{Prop}\ .\ (\mathit{ProofOf}(\mathit{and}\ p\ q\ ) \to \mathit{ProofOf}(\mathit{and}\ q\ p\ ))\ .$$

(iii) What logical proof principle does your term demonstrate? [*8 marks*]

3. (a) The following screen shots of the US health insurance website healthcare.gov circulated widely in the press shortly after its launch in 2013.



These autocompletion prompts represent a record of user attempts to carry out *injection attacks* on the site.

(i) Explain in a sentence or two what an *SQL injection attack* aims to do, and how it works.

(ii) Suppose a malicious user enters the text `'; show tables; --` into a website like this. Give an outline of sample code in Java or C# that would be susceptible such an attack. What would be the result if this particular attack worked?

[*12 marks*]

QUESTION CONTINUES ON NEXT PAGE

(b) The rest of this question is about *metaprogramming*.

  (i) What is metaprogramming?

  (ii) Name two examples of metaprogramming; for each one, explain what it does, and state whether it acts at compile time or run time.

  (iii) Give a short description of *(quasi)quotation* and *antiquotation*, with an example in LISP, MetaOCaml, F#, or other programming language of your choice.

[*13 marks*]