UNIVERSITY OF EDINBURGH

FACULTY OF SCIENCE AND ENGINEERING

COMPUTER SCIENCE THREE

PROGRAMMING METHODOLOGY

Friday 8 September 2000

09:30 to 11:00

Examiners:

Prof. K. Turner (External)

Prof. G. Brebner (Chairman)

INSTRUCTIONS TO CANDIDATES

Answer any TWO questions.

All questions carry equal weight.

1. Your firm is developing a computer-assisted hammer. The hammer will pound in nails while attempting to avoid pounding fingers. It incorporates a metal detector and regards presence of metal as indicating absence of fingers, while absence of metal indicates possible presence of fingers. If triggered while metal is absent, it will do nothing. If triggered again, an electronic voice will ask "Are you sure?" If triggered a third time without sensing metal, it will pound despite the possible presence of fingers — perhaps the user is using the hammer to crush ice.

   A critical subsystem is described by the following signature:

```
signature BASH =
  sig
    type State
    val init : State
    val trigger : State * bool -> State
    val pound : State -> bool
    val warning : State -> bool
  end
```

   The state of the subsystem incorporates the state of the hammer and the warning signal (*pound(s)=true* means the hammer has pounded in response to the last application of *trigger*, and is initially *false*, and similarly for *warning*). The next state is produced by the *trigger* function. This takes the current state as input together with the current value of the metal detector (*true* indicates presence of metal).

   (a) Explain the difference between verification and validation, and between static and dynamic methods. What is stress testing? [*5 marks*]

   (b) Describe two methods for dynamic verification of this subsystem and highlight the differences between them. Give an example test set and justify your choice of tests. [*10 marks*]

   (c) Describe one technique for static verification and indicate what information you would require to apply the technique to this subsystem. [*5 marks*]

   (d) Suppose that a defect is revealed. Describe the activities that are involved in debugging the system. [*5 marks*]

2. The government has decided to convert all job centres to internet cafes. All information about job vacancies will be available online in a single centralized website. Your firm has won the contract to build this website and you have been put in charge of the project.

   Job-seekers visit to look for jobs that match their qualifications and score highly with respect to their wishes concerning salary, location, etc. They may register to be notified by e-mail when such jobs become available in future. Employers visit to advertise vacancies. Information about job-seekers is not available to employers but social security staff have privileged access to enable them to verify jobseekers' claims of actively searching for a job. The system's administrator needs to be able to initialize the system, enable and suspend access, and reconfigure the system while it is running. The system should be capable of coping with access by up to 100000 users simultaneously, and storing information concerning 10 million jobseekers and 10 million vacancies. Information about jobseekers, employers, vacancies etc. will be kept in a database using database management software that you have employed in the past to build smaller-scale systems that are similar. The web site is essentially a front end to enable user-friendly access to that database.

   (a) Give a brief description of the functions that the system might provide to a job-seeker. [*5 marks*]

   (b) Choose three of the functions in your chosen interface and provide a detailed specification of each one. [*5 marks*]

   (c) What attributes do you think would be important in the requirements specification of this system? Justify your choice and suggest appropriate measures of these attributes. [*5 marks*]

   (d) You need to select a lifecycle model to use. Outline two lifecycle models and compare their suitability for developing this system. [*5 marks*]

   (e) Define *risk*. Identify three risks of this project. How might you reduce these risks to manageable levels? [*5 marks*]

3. The following defines an expression as either a variable or the application of an operator (represented as a string) to a list of expressions, with constants regarded as 0-ary operators.

```
datatype exp = app of string * exp list | var of string
```

Now consider the following program for matching one expression against another, producing a substitution. (The intention is that, for example,

```
match( app("f",[var "x",     app("g",[]), var "x"]),
       app("f",[app("h",[]), app("g",[]), app("h",[])]) )
```

is the substitution taking `"x"` to `app("h",[])`.)

```
fun matchexp(app(c,l), app(c',l'), s) =
        if c=c' then matchl(l,l',s) else none
  | matchexp(var x, exp, s) =
        if member(x,s) then if lookup(x,s)=exp then s else none
        else add(x,exp,s)
  | matchexp(app(_,_), var _, _) = none
and matchl(nil,nil,s) = s
  | matchl(a::l,a'::l',s) = matchl(l,l',matchexp(a,a',s))
  | matchl(_,_,s) = none       (* lists of different lengths *)
fun match(e,e') = matchexp(e,e',empty)
```

This assumes a type subst of substitutions with

```
empty : subst
member : string * subst -> bool
lookup : string * subst -> exp
add : string * exp * subst -> subst
none : subst
```

where `add(x,e,none)=none` and `lookup(x,none)` raises an exception.

(a) Write Extended ML axioms specifying as abstractly as possible the behaviour of match. You may wish to use the following auxiliary function that replaces the variables in an expression according to a substitution:

```
fun apply(s,var x) = lookup(x,s)
  | apply(s,app(c,l)) = app(c,map (fn exp => apply(s,exp)) l)
```

[10 marks]

QUESTION CONTINUES ON NEXT PAGE

(b) Consider the following implementation of substitutions:

```
datatype subst = some of (string * exp) list | none
val empty = some []
fun member(x,none) = false
  | member(x,some []) = false
  | member(x,some((y,_)::l)) = (x=y) orelse member(x,some l)
exception Fail
fun lookup(x,none) = raise Fail
  | lookup(x,some []) = var x
  | lookup(x,some((y,exp)::l)) =
        if x=y then exp else lookup(x,some l)
fun add(x,exp,none) = none
  | add(x,exp,some l) = some((x,exp)::l)
```

Show that

```
forall (l,l',x) =>
        lookup(x,some(l@l')) == apply(some l',lookup(x,some l))
```

holds if `l` associates variables with variable-free expressions. The proof need not be completely formal. [*15 marks*]