UNIVERSITY OF EDINBURGH

COLLEGE OF SCIENCE AND ENGINEERING

SCHOOL OF INFORMATICS

TYPES AND SEMANTICS FOR PROGRAMMING LANGUAGES

Saturday 1st April 2017

00:00 to 00:00

INSTRUCTIONS TO CANDIDATES

Answer QUESTION 1 and ONE other question.

Question 1 is COMPULSORY. If both QUESTION 2 and QUESTION 3 are answered, only QUESTION 2 will be marked.

All questions carry equal weight.

CALCULATORS MAY NOT BE USED IN THIS EXAMINATION

Year 4 Courses

Convener: ITO-Will-Determine
External Examiners: ITO-Will-Determine

THIS EXAMINATION WILL BE MARKED ANONYMOUSLY

1. THIS QUESTION IS COMPULSORY

Consider a type of trees defined as follows.

$$\texttt{leaf}\frac{A}{\mathit{Tree}\ A} \qquad \texttt{\_branch\_}\frac{\mathit{Tree}\ A \quad \mathit{Tree}\ A}{\mathit{Tree}\ A}$$

Given a predicate P over A, we define predicates `AllT` and `AnyT` which hold when P holds for *every* leaf in the tree and when P holds for *some* leaf in the tree, respectively.

$$\texttt{leaf}\frac{P\ x}{\texttt{AllT}\ P\ (\texttt{leaf}\ x)} \qquad \texttt{\_branch\_}\frac{\texttt{AllT}\ P\ xt \quad \texttt{AllT}\ P\ yt}{\texttt{AllT}\ P\ (xt\ \texttt{branch}\ yt)}$$

$$\texttt{leaf}\frac{P\ x}{\texttt{AnyT}\ P\ (\texttt{leaf}\ x)} \qquad \texttt{left}\frac{\texttt{AnyT}\ P\ xt}{\texttt{AnyT}\ P\ (xt\ \texttt{branch}\ yt)} \qquad \texttt{right}\frac{\texttt{AnyT}\ P\ yt}{\texttt{AnyT}\ P\ (xt\ \texttt{branch}\ yt)}$$

(a) Formalise the definitions above. [*12 marks*]

(b) Prove $\texttt{AllT}\ (\neg\_ \circ P)\ xt$ implies $\neg\ (\texttt{AnyT}\ P\ xt)$, for all trees xt. [*13 marks*]

2. ANSWER EITHER THIS QUESTION OR QUESTION 3

You will be provided with a definition of intrinsically-typed lambda calculus in Agda. Consider constructs satisfying the following rules, written in extrinsically-typed style.

A computation of type $\texttt{Comp}\ A$ returns either an error with a message *msg* which is a string, or an ok value of a term M of type A. Consider constructs satisfying the following rules:

Typing:

$$\texttt{error}\frac{}{\Gamma \vdash \texttt{error}\ msg ⦂ \texttt{Comp}\ A} \qquad \texttt{ok}\frac{\Gamma \vdash M ⦂ A}{\Gamma \vdash \texttt{ok}\ M ⦂ \texttt{Comp}\ A}$$

$$\texttt{letc}\frac{\begin{array}{c}\Gamma \vdash M ⦂ \texttt{Comp}\ A \\ \Gamma\, ,\, x ⦂ A \vdash N ⦂ \texttt{Comp}\ B\end{array}}{\Gamma \vdash \texttt{letc}\ x \leftarrow M\ \texttt{in}\ N ⦂ \texttt{Comp}\ B}$$

Values:

$$\texttt{V-error}\frac{}{\texttt{Value}\ (\texttt{error}\ msg)} \qquad \texttt{V-ok}\frac{\texttt{Value}\ V}{\texttt{Value}\ (\texttt{ok}\ V)}$$

Reduction:

$$\xi\texttt{-ok}\frac{M \longrightarrow M'}{\texttt{ok}\ M \longrightarrow \texttt{ok}\ M'} \qquad \xi\texttt{-letc}\frac{M \longrightarrow M'}{\texttt{letc}\ x \leftarrow M\ \texttt{in}\ N \longrightarrow \texttt{letc}\ x \leftarrow M'\ \texttt{in}\ N}$$

$$\beta\texttt{-error}\frac{}{\texttt{letc}\ x \leftarrow (\texttt{error}\ msg)\ \texttt{in}\ t \longrightarrow \texttt{error}\ msg}$$

$$\beta\texttt{-ok}\frac{\texttt{Value} V}{\texttt{letc}\ x \leftarrow (\texttt{ok}\ V)\ \texttt{in}\ N \longrightarrow N\ [\ x := V\ ]}$$

(a) Extend the given definition to formalise the evaluation and typing rules, including any other required definitions. [*12 marks*]

(b) Prove progress. You will be provided with a proof of progress for the simply-typed lambda calculus that you may extend. [*13 marks*]

Please delimit any code you add as follows.

```
-- begin
-- end
```

3. ANSWER EITHER THIS QUESTION OR QUESTION 2

You will be provided with a definition of inference for extrinsically-typed lambda calculus in Agda. Consider constructs satisfying the following rules, written in extrinsically-typed style that support bidirectional inference.

Typing:

$$\texttt{tt}\ \frac{}{\Gamma \vdash \texttt{tt} \downarrow \top}$$

$$\texttt{case}\top\ \frac{\Gamma \vdash L \uparrow \top \qquad \Gamma \vdash M \downarrow A}{\Gamma \vdash \texttt{case}\top\ L\ [\texttt{tt} \Rightarrow M\ ] \downarrow A}$$

(a) Extend the given definition to formalise the typing rules, and update the definition of equality on types. [*10 marks*]

(b) Extend the code to support type inference for the new features. [*15 marks*]

Please delimit any code you add as follows.

```
-- begin
-- end
```