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

This question uses the library definition of `list` in Agda. Here is an informal definition of the predicates ∈ and ⊆. (In Emacs, you can type ∈ as `\in` and ⊆ as `\subseteq`.) ⊆

$$\texttt{here}\frac{}{x \in (x :: xs)} \qquad \texttt{there}\frac{x \in ys}{x \in (y :: ys)}$$

$$\texttt{done}\frac{}{\texttt{[]} \subseteq ys}$$

$$\texttt{keep}\frac{xs \subseteq ys}{(x :: xs) \subseteq (x :: ys)} \qquad \texttt{drop}\frac{xs \subseteq ys}{xs \subseteq (y :: ys)}$$

(a) Formalise the definition above. [*10 marks*]

(b) Prove each of the following.

(i) `2 ∈ [1,2,3]`

(ii) `[1,3] ⊆ [1,2,3,4]` [*5 marks*]

(c) Prove the following.

If $xs \subseteq ys$ then $z \in xs$ implies $z \in ys$ for all z. [*10 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.

Typing:

$$\texttt{leaf}\ \frac{\Gamma \vdash M \mathbin{⦂} A}{\Gamma \vdash \texttt{leaf}\ M \mathbin{⦂} \texttt{Tree}\ A} \qquad \texttt{branch}\ \frac{\begin{array}{c}\Gamma \vdash M \mathbin{⦂} \texttt{Tree}\ A \\ \Gamma \vdash N \mathbin{⦂} \texttt{Tree}\ A\end{array}}{\Gamma \vdash M\ \texttt{branch}\ N \mathbin{⦂} \texttt{Tree}\ A}$$

$$\texttt{caseT}\ \frac{\begin{array}{c}\Gamma \vdash L \mathbin{⦂} \texttt{Tree}\ A \\ \Gamma\ ,\ x \mathbin{⦂} A \vdash M \mathbin{⦂} B \\ \Gamma\ ,\ y \mathbin{⦂} \texttt{Tree}\ A\ ,\ z \mathbin{⦂} \texttt{Tree}\ A \vdash N \mathbin{⦂} B\end{array}}{\Gamma \vdash \texttt{case}\ L\ [\texttt{leaf}\ x\ \Rightarrow\ M \mid y\ \texttt{branch}\ z\ \Rightarrow\ N] \mathbin{⦂} B}$$

Values:

$$\texttt{V-leaf}\ \frac{\texttt{Value}\ V}{\texttt{Value}\ (\texttt{leaf}\ V)} \qquad \texttt{V-branch}\ \frac{\begin{array}{c}\texttt{Value}\ V \\ \texttt{Value}\ W\end{array}}{\texttt{Value}\ (V\ \texttt{branch}\ W)}$$

Reduction:

$$\xi\texttt{-leaf}\ \frac{M \longrightarrow M'}{\texttt{leaf}\, M \longrightarrow \texttt{leaf}\, M'}$$

$$\xi\texttt{-branch}_1\ \frac{M \longrightarrow M'}{M\ \texttt{branch}\ N \longrightarrow M'\ \texttt{branch}\ N} \qquad \xi\texttt{-branch}_2\ \frac{\begin{array}{c}\texttt{Value}\ V \\ N \longrightarrow N'\end{array}}{V\ \texttt{branch}\ N \longrightarrow V\ \texttt{branch}\ N'}$$

$$\xi\texttt{-caseT}\ \frac{L \longrightarrow L'}{\begin{array}{c}\texttt{case}\ L\ [\texttt{leaf}\ x\ \Rightarrow\ M \mid y\ \texttt{branch}\ z\ \Rightarrow\ N] \longrightarrow \\ \texttt{case}\ L'\ [\texttt{leaf}\ x\ \Rightarrow\ M \mid y\ \texttt{branch}\ z\ \Rightarrow\ N]\end{array}}$$

$$\beta\texttt{-leaf}\ \frac{\texttt{Value}\ V}{\texttt{case}\ (\texttt{leaf}\ V)\ [\texttt{leaf}\ x\ \Rightarrow\ M \mid y\ \texttt{branch}\ z\ \Rightarrow\ N] \longrightarrow M\ [\ x := V\ ]}$$

$$\beta\texttt{-branch}\ \frac{\begin{array}{c}\texttt{Value}\ V \\ \texttt{Value}\ W\end{array}}{\texttt{case}\ (V\ \texttt{branch}\ W)\ [\texttt{leaf}\ x\ \Rightarrow\ M \mid y\ \texttt{branch}\ z\ \Rightarrow\ N] \longrightarrow N\ [\ y := V\ ]\ [\ z := W\ ]}$$

(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:

$$\texttt{delay}\frac{\Gamma \vdash M \downarrow A}{\Gamma \vdash \texttt{delay}\ M \downarrow \texttt{Lift}\ A}$$

$$\texttt{force}\frac{\Gamma \vdash L \uparrow \texttt{Lift}\ A}{\Gamma \vdash \texttt{force}\ L \uparrow A}$$

(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
```