**A36892**

Calculators may be used in this examination provided they are not capable of being used to store alphabetical information other than hexadecimal numbers

UNIVERSITY OF BIRMINGHAM

**School of Computer Science**

Third Year Undergraduate

**06-30230**

**LH Programming Language Principles, Design, and Implementation**

Main Summer Examinations 2025

Time allowed: 2 hours

[Answer all questions]



Turn Over

## Note

Answer ALL questions. There are 4 questions. Each question is worth 25 marks.

## Question 1 [$\lambda$-Calculus and type systems – 25 marks]

Consider the following term $M$ of the Church-style simply-typed $\lambda$-calculus.

$$M = \lambda f : \mathbb{B} \to \mathbb{B}.\lambda g : \mathbb{B} \to \mathbb{B} \to \mathbb{B}.\lambda x : \mathbb{B}.\lambda y : \mathbb{B}.\lambda z : \mathbb{B}.\texttt{if } g\, y\, z \texttt{ then } z \texttt{ else } f\, x$$

(a) The following is a partial picture of a typing derivation for $M$. Find the derivation $\Pi$ that completes the top of the derivation, and hence find the missing types $T_1$ and $T_0$.

$$\dfrac{\dfrac{\dfrac{\Pi}{f : \mathbb{B} \to \mathbb{B}, g : \mathbb{B} \to \mathbb{B} \to \mathbb{B}, x : \mathbb{B}, y : \mathbb{B} \vdash \lambda z : \mathbb{B}.\texttt{if } g\, y\, z \texttt{ then } z \texttt{ else } f\, x : T_1}\,\textsc{abs}}{\vdots}\,\textsc{abs}}{\vdash M : T_0}\,\textsc{abs}$$

[Hint: Set an abbreviation $\Gamma$ for the context $f : \mathbb{B} \to \mathbb{B}, g : \mathbb{B} \to \mathbb{B} \to \mathbb{B}, x : \mathbb{B}, y : \mathbb{B}, z : \mathbb{B}$.]

**[6 marks]**

(b) Find well-typed terms $F, G$, such that, for $b_1, b_2, b_3 \in \{\texttt{true}, \texttt{false}\}$,

$$M\, F\, G\, b_1\, b_2\, b_3$$

computes to `true` iff $b_1, b_2, b_3$ are in increasing order. To clarify, 'increasing order' means that

$$(b_1, b_2, b_3) \in \{(\texttt{false}, \texttt{false}, \texttt{false}), (\texttt{false}, \texttt{false}, \texttt{true}), (\texttt{false}, \texttt{true}, \texttt{true}), (\texttt{true}, \texttt{true}, \texttt{true})\}.$$

Prove that your example $(F, G)$ works by formally computing the small-step CBV semantics of $M\, F\, G\, b_1\, b_2\, b_3$. You may save work by combining several consecutive $\beta$-reduction steps if your explanation is clear. [Hint: You should not need to case-split on the values of $b_1, b_2, b_3$ until the last few steps of the derivation.] **[10 marks]**

Recall that the (type) *erasure* of a term is the Curry-style term obtained by removing the type-annotations on binding occurrences of variables. For instance,

$$\texttt{erase}(M) = \lambda f.\lambda g.\lambda x.\lambda y.\lambda z.\texttt{if } g\, y\, z \texttt{ then } z \texttt{ else } f\, x.$$

(c) Suppose that an error is made while copying down $\texttt{erase}(M)$ and we end up with

$$N = \lambda f.\lambda g.\lambda x.\lambda y.\lambda z.\texttt{if } g\, y \texttt{ then } g\, y\, z \texttt{ else } f\, x.$$

By considering a partial derivation tree, show that the term $N$ is not typeable. [You may combine several ABS steps into one, as in part (a).] **[9 marks]**

## Question 2 [Polymorphism and abstraction – 25 marks]

Without using the base type $\mathbb{N}$, we can encode natural numbers in System F as the type

$$\mathtt{Nat}_* = \forall\alpha.\alpha \to (\alpha \to \alpha) \to \alpha$$

along with

$$\begin{aligned}\mathtt{zero}_* &= \lambda\alpha.\lambda x : \alpha.\lambda f : \alpha \to \alpha.x \\ \mathtt{succ}_* &= \lambda n : \mathtt{Nat}_*.\lambda\alpha.\lambda x : \alpha.\lambda f : \alpha \to \alpha.f\,(n\{\alpha\}\,x\,f)\end{aligned}$$

(a) Prove that $\mathtt{succ}_*$ has type $\mathtt{Nat}_* \to \mathtt{Nat}_*$ using the typing rules of System F. **[8 marks]**

(b) Give a term $R$ of type $\mathtt{Nat}_* \to \mathbb{N}$ which translates from the encoding of natural numbers to the base type $\mathbb{N}$. You do not need to give the typing derivation. Your term should satisfy

$$\begin{aligned}R\,\mathtt{zero}_* &\to_v^* \mathtt{zero} \\ R\,(\mathtt{succ}_*\,\mathtt{zero}_*) &\to_v^* \mathtt{succ\ zero}\end{aligned}$$

and so on. Prove carefully that your term does indeed satisfy $R\,(\mathtt{succ}_*\,\mathtt{zero}_*) \to_v^* \mathtt{succ\ zero}$ **[9 marks]**

(c) In System F extended with existential types (System E), find all types T such that

$$\mathtt{pack}\ \langle \mathtt{Nat}_*, \langle \mathtt{zero}_*, R\rangle\rangle\ \mathtt{as}\ \exists\alpha.T$$

is typable. You should explain your answer, but you do not need to give detailed typing derivations. **[8 marks]**

## Question 3 [Parsing – 25 marks]

Consider the following language for bitwise operations in Verilog:

$$\begin{array}{lll} A & ::= & \sim A \mid A \,\&\, A \mid A \mid A \mid A \,\hat{}\, A \mid B \mid (A) \\ B & ::= & 0 \mid 1 \mid BB \end{array}$$

The meaning of these expressions is not important for this question, but informally, a $B$ expression is a list of bits, $\sim A$ is the bitwise inverse of $A$, $A_1 \mid A_2$ is the bitwise or operation between $A_1$ and $A_2$, $A_1 \,\&\, A_2$ is the bitwise and operation between $A_1$ and $A_2$, and $A_1 \,\hat{}\, A_2$ is the bitwise xor operation between $A_1$ and $A_2$.

(a) Describe one way in which this grammar is ambiguous. Then, provide a modified grammar that fixes this ambiguity, as well as a justification explaining how it fixes it. [Hint: There are multiple ways in which this grammar is ambiguous.]

**[5 marks]**

(b) Write a bottom-up parser for the original grammar as a PDA (a pushdown automaton). Indicate what the initial stack symbol is. [Hint: It might help to think about what transitions are shift transitions and what transitions are reduce transitions. You are allowed to make use of some notation to for example pop two elements from the stack, but if you do, briefly explain your notation.]

**[10 marks]**

(c) Show that the expression $\sim 0 \,\hat{}\, 1$ is accepted by your PDA by providing a sequence of instance descriptions that details all the steps. [Hint: It might again help to think about what steps are shift steps and what steps are reduce steps.]

**[5 marks]**

(d) Explain where a shift/reduce ambiguity arises in your sequence of instance descriptions, and what the implications are of the decision you have made (shift vs. reduce) on how the expression is parsed, comparing with what would have happened had you made another decision. [Hint: A shift/reduce ambiguity is when either a shift or a reduce can be performed on the same instance description, while still getting the word to be accepted.]

**[5 marks]**

## Question 4 [Code Generation & Optimization – 25 marks]

(a) A Boolean expression of the form $B_1$ || $B_2$ (a disjunction) is translated to TAC code as follows: $\texttt{tac}(B_1 \texttt{ || } B_2, L_1, L_2) = \texttt{tac}(B_1, L_1, L') \parallel L' \parallel \texttt{tac}(B_2, L_1, L_2)$, where $L'$ is a new label, and $\parallel$ concatenates two sequences of TAC instructions. Extend the `tac` function to Boolean expressions of the form $B_1 \uparrow B_2$, where $\uparrow$ is the *nand* Boolean operator, which returns false iff both $B_1$ and $B_2$ are true, i.e., explain what $\texttt{tac}(B_1 \uparrow B_2, L_1, L_2)$ returns.

**[5 marks]**

(b) Provide the code generated by this extended `tac` function given the Boolean expression `true` $\uparrow$ `false`, and explain how that code could be optimised.

**[5 marks]**

(c) Consider the following basic block that contains 2 TAC instructions:

$$a = a + b$$
$$a = a + b$$

Translate the above block to assembly code one instruction at a time, using at most 3 registers, showing how the register and address descriptors are updated, and assuming that $a$ and $b$ initially hold their respective values.

**[10 marks]**

(d) Optimize the above TAC code assuming that $a$ and $b$ are used in other blocks, and compare the cost of the generated code before and after optimizing the TAC code. In particular, explain whether common sub-expression elimination can be used to optimized the TAC code.

**[5 marks]**

## End of Questions – See below for the appendix

# Appendix

## 1 Church-Style Simply Typed $\lambda$-calculus

### Syntax

$$\begin{array}{lcl} T & ::= & \mathbb{B} \mid T \to T \\ M & ::= & x \mid \lambda x : T.\, M \mid M\ M \mid \texttt{true} \mid \texttt{false} \mid \texttt{if}\ M\ \texttt{then}\ M\ \texttt{else}\ M \end{array}$$

### Values

$$V ::= \lambda x : T.\, M \mid \texttt{true} \mid \texttt{false}$$

### Call-by-value evaluation contexts

$$C ::= \bullet \mid C\ M \mid V\ C \mid \texttt{if}\ C\ \texttt{then}\ M\ \texttt{else}\ M$$

### Call-by-value small-step operational semantics rules

$$\frac{}{(\lambda x : T.\, M)\ V \longrightarrow_v M[x \backslash V]}\ ^{\beta} \qquad \frac{}{\texttt{if true then}\ M\ \texttt{else}\ N \longrightarrow_v M}\ ^{\texttt{IteT}}$$

$$\frac{}{\texttt{if false then}\ M\ \texttt{else}\ N \longrightarrow_v N}\ ^{\texttt{IteF}} \qquad \frac{M \longrightarrow_v N}{C[M] \longrightarrow_v C[N]}\ ^{\texttt{CTX}_C}$$

### Typing Rules

$$\frac{}{\Gamma, x : T \vdash x : T}\ ^{\texttt{VAR}} \qquad \frac{\Gamma, x : T \vdash M : U}{\Gamma \vdash \lambda x : T.\, M : T \to U}\ ^{\texttt{ABS}} \qquad \frac{\Gamma \vdash M : T \to U \quad \Gamma \vdash N : T}{\Gamma \vdash M\ N : U}\ ^{\texttt{APP}}$$

$$\frac{}{\Gamma \vdash \texttt{true} : \mathbb{B}}\ ^{\texttt{T}} \qquad \frac{}{\Gamma \vdash \texttt{false} : \mathbb{B}}\ ^{\texttt{F}} \qquad \frac{\Gamma \vdash M : \mathbb{B} \quad \Gamma \vdash N : T \quad \Gamma \vdash P : T}{\Gamma \vdash \texttt{if}\ M\ \texttt{then}\ N\ \texttt{else}\ P : T}\ ^{\texttt{ITE}}$$

## 2 System F

### Syntax

$$\begin{array}{lcl} T & ::= & \alpha \mid \mathbb{B} \mid \mathbb{N} \mid T \to T \mid \forall \alpha.T \\ M & ::= & x \mid \lambda x : T.\, M \mid M\ M \mid \texttt{true} \mid \texttt{false} \mid \texttt{if}\ M\ \texttt{then}\ M\ \texttt{else}\ M \\ & \mid & \texttt{let}\ x\ =\ M\ \texttt{in}\ M \mid \texttt{zero} \mid \texttt{succ}\ M \mid \texttt{pred}\ M \mid \texttt{iszero}\ M \mid \lambda \alpha.\, M \mid M\{T\} \end{array}$$

### Values

$$V ::= \lambda x : T.\, M \mid \texttt{true} \mid \texttt{false} \mid \texttt{zero} \mid \texttt{succ}\ V \mid \lambda \alpha.\, M$$

### Call-by-value evaluation contexts

$$\begin{array}{rcl} C & ::= & \bullet \mid C\ M \mid V\ C \mid \texttt{if}\ C\ \texttt{then}\ M\ \texttt{else}\ M \\ & \mid & \texttt{let}\ x\ =\ C\ \texttt{in}\ M \mid \texttt{pred}\ C \mid \texttt{iszero}\ C \mid \texttt{succ}\ C \mid C\{T\} \end{array}$$

### Call-by-value small-step operational semantics rules

$$\frac{}{(\lambda x : T.\, M)\ V \longrightarrow_v M[x \backslash V]}\ \beta \qquad \frac{M \longrightarrow_v N}{C[M] \longrightarrow_v C[N]}\ \text{CTX}_C$$

$$\frac{}{\texttt{if true then}\ M\ \texttt{else}\ N \longrightarrow_v M}\ \text{IteT} \qquad \frac{}{\texttt{if false then}\ M\ \texttt{else}\ N \longrightarrow_v N}\ \text{IteF}$$

$$\frac{}{\texttt{pred zero} \longrightarrow_v \texttt{zero}}\ \text{PredZ} \qquad \frac{}{\texttt{pred (succ}\ V) \longrightarrow_v V}\ \text{PredS}$$

$$\frac{}{\texttt{iszero zero} \longrightarrow_v \texttt{true}}\ \text{IsZZ} \qquad \frac{}{\texttt{iszero (succ}\ V) \longrightarrow_v \texttt{false}}\ \text{IsZS}$$

$$\frac{}{\texttt{let}\ x\ =\ V\ \texttt{in}\ M \longrightarrow_v M[x \backslash V]}\ \text{LetV} \qquad \frac{}{(\lambda \alpha.\, M)\{T\} \longrightarrow_v M[\alpha \backslash T]}\ \text{T}\beta$$

### Typing rules

$$\frac{}{\Gamma, x : T \vdash x : T}\ \text{VAR} \qquad \frac{\Gamma, x : T \vdash M : U}{\Gamma \vdash \lambda x : T.\, M : T \to U}\ \text{ABS} \qquad \frac{\Gamma \vdash M : T \to U \quad \Gamma \vdash N : T}{\Gamma \vdash M\ N : U}\ \text{APP}$$

$$\frac{}{\Gamma \vdash \texttt{true} : \mathbb{B}}\ \text{T} \qquad \frac{}{\Gamma \vdash \texttt{false} : \mathbb{B}}\ \text{F} \qquad \frac{\Gamma \vdash M : \mathbb{B} \quad \Gamma \vdash N : T \quad \Gamma \vdash P : T}{\Gamma \vdash \texttt{if}\ M\ \texttt{then}\ N\ \texttt{else}\ P : T}\ \text{ITE}$$

$$\frac{}{\Gamma \vdash \texttt{zero} : \mathbb{N}}\ \text{Z} \qquad \frac{\Gamma \vdash M : \mathbb{N}}{\Gamma \vdash \texttt{succ}\ M : \mathbb{N}}\ \text{S} \qquad \frac{\Gamma \vdash M : \mathbb{N}}{\Gamma \vdash \texttt{pred}\ M : \mathbb{N}}\ \text{S} \qquad \frac{\Gamma \vdash M : \mathbb{N}}{\Gamma \vdash \texttt{iszero}\ M : \mathbb{B}}\ \text{I}$$

$$\frac{\Gamma \vdash M : T \quad \Gamma, x : T \vdash N : U}{\Gamma \vdash \texttt{let}\ x\ =\ M\ \texttt{in}\ N : U}\ \text{LET} \qquad \frac{\Gamma \vdash M : T}{\Gamma \vdash \lambda \alpha.\, M : \forall \alpha.T}\ \text{TABS} \qquad \frac{\Gamma \vdash M : \forall \alpha.T}{\Gamma \vdash M\{U\} : T[\alpha \backslash U]}\ \text{TAPP}$$

In addition TABS requires that $\alpha \notin \texttt{FV}(\Gamma)$.

# 3 System E

### Syntax

$$\begin{array}{rcl} T & ::= & \alpha \mid \mathbb{B} \mid \mathbb{N} \mid T \to T \mid \forall \alpha.T \mid T \times T \mid \exists \alpha.T \\ M & ::= & x \mid \lambda x : T.\, M \mid M\ M \mid \texttt{true} \mid \texttt{false} \mid \texttt{if}\ M\ \texttt{then}\ M\ \texttt{else}\ M \\ & \mid & \texttt{let}\ x\ =\ M\ \texttt{in}\ M \mid \texttt{zero} \mid \texttt{succ}\ M \mid \texttt{pred}\ M \mid \texttt{iszero}\ M \\ & \mid & \lambda \alpha.\, M \mid M\{T\} \\ & \mid & \langle M, M \rangle \mid \texttt{fst}\ M \mid \texttt{snd}\ M \\ & \mid & \texttt{pack}\ \langle T, M \rangle\ \texttt{as}\ T \mid \texttt{unpack}\ \alpha, x\ =\ M\ \texttt{in}\ M \end{array}$$

**Values**

$$\begin{array}{rcl} V & ::= & \lambda x : T.\, M \mid \texttt{true} \mid \texttt{false} \mid \texttt{zero} \mid \texttt{succ}\ V \mid \lambda\alpha.\, M \\ & \mid & \langle V, V \rangle \mid \texttt{pack}\ \langle T, V \rangle\ \texttt{as}\ T \end{array}$$

**Call-by-value evaluation contexts**

$$\begin{array}{rcl} C & ::= & \bullet \mid C\ M \mid V\ C \mid \texttt{if}\ C\ \texttt{then}\ M\ \texttt{else}\ M \\ & \mid & \texttt{let}\ x\ =\ C\ \texttt{in}\ M \mid \texttt{pred}\ C \mid \texttt{iszero}\ C \mid \texttt{succ}\ C \\ & \mid & C\{T\} \mid \texttt{unpack}\ \alpha, x\ =\ C\ \texttt{in}\ M \mid \texttt{fst}\ C \mid \texttt{snd}\ C \\ & \mid & \langle C, M \rangle \mid \langle V, C \rangle \mid \texttt{pack}\ \langle T, C \rangle\ \texttt{as}\ T \end{array}$$

**Call-by-value small-step operational semantics rules**

$$\frac{}{(\lambda x : T.\, M)\ V \longrightarrow_v M[x \backslash V]}\ \beta \qquad \frac{M \longrightarrow_v N}{C[M] \longrightarrow_v C[N]}\ \text{CTX}_C$$

$$\frac{}{\texttt{if true then}\ M\ \texttt{else}\ N \longrightarrow_v M}\ \text{IteT} \qquad \frac{}{\texttt{if false then}\ M\ \texttt{else}\ N \longrightarrow_v N}\ \text{IteF}$$

$$\frac{}{\texttt{let}\ x\ =\ V\ \texttt{in}\ M \longrightarrow_v M[x \backslash V]}\ \text{LetV} \qquad \frac{}{(\lambda\alpha.\, M)\{T\} \longrightarrow_v M[\alpha \backslash T]}\ \text{T}\beta$$

$$\frac{}{\texttt{pred zero} \longrightarrow_v \texttt{zero}}\ \text{PredZ} \qquad \frac{}{\texttt{pred}\ (\texttt{succ}\ V) \longrightarrow_v V}\ \text{PredS}$$

$$\frac{}{\texttt{iszero zero} \longrightarrow_v \texttt{true}}\ \text{IsZZ} \qquad \frac{}{\texttt{iszero}\ (\texttt{succ}\ V) \longrightarrow_v \texttt{false}}\ \text{IsZS}$$

$$\frac{}{\texttt{fst}\ \langle V, W \rangle \longrightarrow_v V}\ \text{FstP} \qquad \frac{}{\texttt{snd}\ \langle V, W \rangle \longrightarrow_v W}\ \text{SndP}$$

$$\frac{}{\texttt{unpack}\ \alpha, x\ =\ \texttt{pack}\ \langle T, V \rangle\ \texttt{as}\ U\ \texttt{in}\ N \longrightarrow_v N[\alpha \backslash T][x \backslash V]}\ \text{UnP}$$

**Typing rules**

$$\frac{}{\Gamma, x : T \vdash x : T}\ \textsc{var} \qquad \frac{\Gamma, x : T \vdash M : U}{\Gamma \vdash \lambda x : T.\, M : T \to U}\ \textsc{abs} \qquad \frac{\Gamma \vdash M : T \to U \quad \Gamma \vdash N : T}{\Gamma \vdash M\ N : U}\ \textsc{app}$$

$$\frac{}{\Gamma \vdash \texttt{true} : \mathbb{B}}\ \textsc{t} \qquad \frac{}{\Gamma \vdash \texttt{false} : \mathbb{B}}\ \textsc{f} \qquad \frac{\Gamma \vdash M : \mathbb{B} \quad \Gamma \vdash N : T \quad \Gamma \vdash P : T}{\Gamma \vdash \texttt{if}\ M\ \texttt{then}\ N\ \texttt{else}\ P : T}\ \textsc{ite}$$

$$\frac{}{\Gamma \vdash \texttt{zero} : \mathbb{N}}\ \textsc{z} \qquad \frac{\Gamma \vdash M : \mathbb{N}}{\Gamma \vdash \texttt{succ}\ M : \mathbb{N}}\ \textsc{s} \qquad \frac{\Gamma \vdash M : \mathbb{N}}{\Gamma \vdash \texttt{pred}\ M : \mathbb{N}}\ \textsc{s} \qquad \frac{\Gamma \vdash M : \mathbb{N}}{\Gamma \vdash \texttt{iszero}\ M : \mathbb{B}}\ \textsc{i}$$

$$\frac{\Gamma \vdash M : T \quad \Gamma, x : T \vdash N : U}{\Gamma \vdash \texttt{let}\ x\ =\ M\ \texttt{in}\ N : U}\ \textsc{let} \qquad \frac{\Gamma \vdash M : T}{\Gamma \vdash \lambda\alpha.\, M : \forall\alpha.T}\ \textsc{tabs} \qquad \frac{\Gamma \vdash M : \forall\alpha.T}{\Gamma \vdash M\{U\} : T[\alpha \backslash U]}\ \textsc{tapp}$$

$$\frac{\Gamma \vdash M : T \quad \Gamma \vdash N : U}{\Gamma \vdash \langle M, N \rangle : T \times U}\ \textsc{Pair} \qquad \frac{\Gamma \vdash M : T \times U}{\Gamma \vdash \texttt{fst}\ M : T}\ \textsc{Fst} \qquad \frac{\Gamma \vdash M : T \times U}{\Gamma \vdash \texttt{snd}\ M : U}\ \textsc{Snd}$$

$$\frac{\Gamma \vdash M : U[\alpha \backslash T]}{\Gamma \vdash \texttt{pack}\ \langle T, M \rangle\ \texttt{as}\ \exists\alpha.U : \exists\alpha.U}\ \textsc{Pack} \qquad \frac{\Gamma \vdash M : \exists\alpha.T \quad \Gamma, x : T \vdash N : U}{\Gamma \vdash \texttt{unpack}\ \alpha, x\ =\ M\ \texttt{in}\ N : U}\ \textsc{Unpack}$$

In addition TABS requires that $\alpha \notin \texttt{FV}(\Gamma)$
and Unpack requires that $\alpha \notin \texttt{FV}(U)$

A36892 LH Programming Language Principles, Design, and Implementation Non-alpha only

**Do not complete the attendance slip, fill in the front of the answer book or turn over the question paper until you are told to do so**

**Important Reminders**

- Coats/outwear should be placed in the designated area.
- Unauthorised materials (e.g. notes or Tippex) must be placed in the designated area.
- Check that you do not have any unauthorised materials with you (e.g. in your pockets, pencil case).
- Mobile phones and smart watches **must** be switched off and placed in the designated area or under your desk. They must not be left on your person or in your pockets.
- You are not permitted to use a mobile phone as a clock. If you have difficulty seeing a clock, please alert an Invigilator.
- You are not permitted to have writing on your hand, arm or other body part.
- Check that you do not have writing on your hand, arm or other body part – if you do, you must inform an Invigilator immediately
- Alert an Invigilator immediately if you find any unauthorised item upon you during the examination.

**Any students found with non-permitted items upon their person during the examination, or who fail to comply with Examination rules may be subject to Student Conduct procedures.**