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

Programming Language Principles, Design and Implementation

Main Summer Examinations 2024

Time allowed: 2 hours

[Answer all questions]

Turn Over

Note

Answer ALL questions. Each question will be marked out of 20. The paper will be marked out of 80, which will be rescaled to a mark out of 100.

Question 1 [λ-Calculus and type systems]

Consider the following extension of the Church-style Simply-Typed λ-Calculus, with injections and disjoint unions as primitives:

$$\begin{array}{rcl} M & ::= & x \mid \lambda x : T.\, M \mid M\ M \mid \texttt{true} \mid \texttt{false} \mid \texttt{if}\ M\ \texttt{then}\ M\ \texttt{else}\ M \\ & \mid & \texttt{left}(M) \mid \texttt{right}(M) \mid \texttt{case}(M, M, M) \\ T & ::= & \mathbb{B} \mid T \to T \mid T + T \end{array}$$

Expressions of the form $\texttt{left}(M)$ and $\texttt{right}(N)$ are injections, which are values of the disjoint union type $T_1 + T_2$ whenever M is a value of type T_1 and N is a value of type T_2. The $\texttt{case}(M, P, Q)$ operator, is used to pattern match on injections, and takes an M of type $T_1 + T_2$, a P of type $T_1 \to U$, and a Q of type $T_2 \to U$. It computes as follows: it reduces M to an injection: either of the form $\texttt{left}(V)$, where V is a value, and returns $P\ V$, or of the form $\texttt{right}(V)$ and returns $Q\ V$.

(a) Extend the call-by-value small-step operational semantics of the Church-style Simply Typed λ-Calculus to account for the new forms mentioned above, namely: $\texttt{left}(M)$, $\texttt{right}(M)$ and $\texttt{case}(M, P, Q)$. To do this you need to extend its terms, values, evaluation contexts, and operational semantics rules and axioms.

[6 marks]

(b) Extend the typing rules of the Church-style Simply Typed λ-calculus with these expressions.

[4 marks]

(c) Prove that $\texttt{case}(\texttt{left}(\texttt{true}), \lambda x : \mathbb{B}.\, \texttt{if}\ x\ \texttt{then false else true}, \lambda x : \mathbb{B}.\, x)$ evaluates to a value. Detail all the steps.

[6 marks]

(d) Can we find an expression M such that $\texttt{case}(M, \lambda x : \mathbb{B}.\, x, \lambda x : \mathbb{B}.\, x)$ gets stuck? Justify your answer with one or two sentences.

[4 marks]

Question 2 [Polymorphism and abstraction]

The disjoint union type $T_1 + T_2$ is a type with two constructors: a left injection that creates an element of type $T_1 + T_2$ out of an element of type T_1; and a right injection that creates an element of type $T_1 + T_2$ out of an element of type T_2. The disjoint union can be encoded in System F using a Church encoding as follows: $\forall\alpha.(T_1 \to \alpha) \to (T_2 \to \alpha) \to \alpha$. The left injection building a term of type $T_1 + T_2$ out of a term of type T_1 and the right injection building a term of type $T_1 + T_2$ out of a term of type T_2 can then be encoded as follows:

- $\texttt{left} = \lambda x : T_1.\, \lambda\alpha.\, \lambda f : T_1 \to \alpha.\, \lambda g : T_2 \to \alpha.\, f\ x$
- $\texttt{right} = \lambda x : T_2.\, \lambda\alpha.\, \lambda f : T_1 \to \alpha.\, \lambda g : T_2 \to \alpha.\, g\ x$

(a) Prove that `left` is of type $T_1 \to (T_1 + T_2)$ using the typing rules of System F.

[5 marks]

(b) Using the above encoding of disjoint unions, define a `case` function of type

$$\forall\alpha.(T_1 + T_2) \to (T_1 \to \alpha) \to (T_2 \to \alpha) \to \alpha$$

that, depending on whether its argument of type $T_1 + T_2$ encodes a left or right injection, either applies the function of type $T_1 \to \alpha$ (in case of a left injection) or the function of type $T_2 \to \alpha$ (in case of a right injection).

[4 marks]

(c) Explain whether the following expression is a well-typed expression of System F:

$$\lambda x : T_1 + T_2.\, \lambda f : T_1 \to T_1.\, x\{T_1\}\ f\ f$$

Use a complete (in case the expression is well-typed) or incomplete (in case the expression is not well-typed) proof tree to justify your answer.

[6 marks]

(d) Define an abstract datatype for disjoint unions $T_1 + T_2$, forming an abstraction barrier around the above encoding of disjoint unions, whose interface provides three operations: a left injection (`left`), a right injection (`right`), and pattern-matching on an element on that type (`case`). You can either use the extension of System F with existential types, or a Haskell module.

[5 marks]

Question 3 [Parsing]

Consider the following language to manipulate lists of bits:

$$\begin{array}{lcl} L & ::= & \texttt{nil} \mid B :: L \mid L +\!\!+ L \mid L \setminus L \mid (L) \\ B & ::= & \texttt{0} \mid \texttt{1} \end{array}$$

The meaning of these expressions is not important for this question, but informally, `nil` is an empty list, $B :: L$ prepends the bit B to the list L, $L_1 +\!\!+ L_2$ appends the list L_2 to the list L_1, and $L_1 \setminus L_2$ removes the bits in the list L_2 from the list L_1. *(Feel free to write n instead of* `nil` *in your answers.)*

(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. *(There are multiple ways in which this grammar is ambiguous.)*

[6 marks]

(b) Write a bottom-up parser for the original grammar as a PDA (a pushdown automaton). Indicate what the initial stack symbol is. *(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, explain your notation.)*

[8 marks]

(c) Show that the expression $\texttt{nil} \setminus \texttt{nil} +\!\!+ \texttt{nil}$ is accepted by your PDA by providing a sequence of instance descriptions that details all the steps. *(It might again help to think about what steps are shift steps and what steps are reduce steps.)*

[4 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. *(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.)*

[2 marks]

Question 4 [Code Generation & Optimization]

Consider the following basic block that contains 4 TAC instructions:

$$
\begin{aligned}
b &= a \\
c &= d + b \\
b &= d \\
e &= b + a
\end{aligned}
$$

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

[10 marks]

(b) Optimize the above TAC code, assuming that a, b, c, d, and e are used in other blocks, mentioning the names of the optimizations you are using, and showing the code you obtain after each optimization step.

[6 marks]

(c) Describe the cost of the optimized TAC code and compare it to the cost of the non-optimized TAC code (no need to translate the optimized code to assembly too). Which version is the less costly? Justify your answer.

[4 marks]

End of Questions – See below for the appendix

Appendix

1 Church-Style Simply Typed λ-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 $\alpha \notin \mathtt{FV}(\Gamma)$
and Unpack requires that $\alpha \notin \mathtt{FV}(U)$

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.