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

LH Programming Language Principles, Design, and Implementation

Main Summer Examinations 2023

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]

(a) Define a variant of the call-by-value λ-calculus, in which the evalutation happens right-to-left instead of left-to-right. That means that, in function applications, the argument term is evaluated first, and the function term next.

You need to give the syntax and operational semantics for the language. Use evaluation contexts to do this. **[6 marks]**

(b) Do the evaluations in this language give different results from the usual call-by-value λ-calculus that evaluates left to right? Explain your answer. **[2 marks]**

(c) Add to the language of part (a), a `let`-construct for local definitions. It should be of the form

$$\texttt{let}\ x\ =\ M\ \texttt{in}\ M$$

Include its operational semantics, along with evaluation contexts. **[4 marks]**

(d) Compare the term $\texttt{let}\ x\ =\ M\ \texttt{in}\ N$ to the term $(\lambda x.N)\ M$, stating the similarities and differences between the two terms in how they evaluate. **[2 marks]**

(e) Assuming that we place a Curry-style *simple type system* on the call-by-value λ-calculus, give the typing rule that should be used for the new `let`-construct. **[3 marks]**

(f) Extend the typing rule for the `let`-construct to allow *polymorphic functions* in the local definition. **[3 marks]**

Question 2 [Polymorphism and abstraction]

(a) Recall the System F studied in the course (Polymorphic Lambda Calculus) along with added types for Booleans ($\mathbb{B}$) and natural numbers ($\mathbb{N}$).

Define a polymorphic function `twice` with type $\forall\alpha.(\alpha \to \alpha) \to (\alpha \to \alpha)$ that applies a given function twice. **[2 marks]**

(b) Show the complete typing derivation for the `twice` function you defined. **[6 marks]**

(c) Use the `twice` function to define a function to increment a natural number by 2. **[2 marks]**

(d) Define a Haskell module for an abstract data type of queues, in which queues are represented as *pairs of lists*.

```
module Queue(Queue,empty,enqueue,dequeue,peek) where
data Queue a = T ([a], [a])

empty :: Queue a
enqueue :: a -> Queue a -> Queue a
dequeue :: Queue a -> Queue a
peek :: Queue a -> Maybe a
...
```

The idea is for the first list to contain some of the queue elements that occur at the *front* of the queue and the second list to contain the remaining elements at the *rear*. The rear elements are stored in the opposite order to those of the front. For example, if the elements 1, 2, 3, 4, 5, 6 were enqueued, and two elements dequeued in the midst of those operations, the two lists might look like:

```
([3, 4], [6, 5])
```

This allows both `enqueue` and `dequeue` to be performed efficiently. However, when the front list is empty and the rear list is nonempty, this representation requires a rearrangement operation whereby the rear elements are moved to the front list, while reversing their order.

Define a private function

```
rearrange: Queue a -> Queue a
```

which rearranges the queue by moving the *rear* elements to *front*, but only if the *front* list is empty. Ideally, this function should work in $O(n)$ time, where n is the number of elements. **[4 marks]**

(e) Define the module functions empty, enqueue, dequeue, and peek. They should make use of the `rearrange` function appropriately so that all the operations have an amortised cost of constant time. (That means that every element should participate in the rearrangement operation at most once.) **[6 marks]**

Question 3 [Parsing]

Consider the following grammar for Boolean expressions: $B ::= \mathtt{T} \mid \mathtt{F} \mid B \ \& \ B \mid \neg B \mid (B)$.

(a) Describe all the ways in which this grammar is ambiguous. **[4 marks]**

(b) Explain where some ambiguity issue comes up when parsing the expression $\neg\mathtt{T} \ \& \ \mathtt{T}$ using the high-level Bottom-Up parsing algorithm seen in the lectures. Your run of the algorithm should be presented as a table that indicates how the stack and input evolve according to the shift and reduce actions. **[8 marks]**

(c) Provide a non-ambiguous grammar that captures the same language and justify your answer. **[8 marks]**

Question 4 [Code Generation & Optimization]

Consider the following basic block that contains 3 instructions:

$$a = b$$
$$d = c + b$$
$$c = c + a$$

(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. **[8 marks]**

(b) Compute the cost of the generated assembly code. Detail your answer, explaining the cost of each instruction. **[4 marks]**

(c) Optimize the above code, assuming that a, b, c, d are used in other blocks, mentioning the names of the optimizations you have used, and describe the cost of the optimized code compared to the cost of the non-optimized code. **[8 marks]**

End of Paper

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.