Programming Language Concepts

Final Examination, II Semester, 2023–2024

Date : 4 May, 2024
Duration : 3 hours

Marks : 40
Weightage : 40%

- *Answer all questions.*
- *In any question that asks for code, provide Java or Rust style pseudocode, as appropriate. Syntax errors will be ignored, provided the code is conceptually correct.*
- *Supply explanations for any code you write, ideally as annotations alongside the code.*
- *You can assume the following standard encodings introduced in the lecture slides, with the appropriate behaviour:*

$f^0 x$	:=	x	$f^{i+1} x$	:=	$f(f^i x)$
$\langle n \rangle$	:=	$\lambda fx \cdot f^n x$	**succ**	:=	$\lambda pfx \cdot f(pfx)$
plus	:=	$\lambda pqfx \cdot pf(qfx)$	**pair**	:=	$\lambda xyw \cdot wxy$
fst	:=	$\lambda p \cdot p$ **true**	**snd**	:=	$\lambda p \cdot p$ **false**
true	:=	$\lambda xy \cdot x$	**false**	:=	$\lambda xy \cdot y$
ite	:=	$\lambda bxy \cdot bxy$	**iszero**	:=	$\lambda x \cdot (x(\lambda z \cdot$ **false**$))$ **true**

---

1. Assume that you have available a method

```
public static int System.in.readPrime()
    throws NotIntegerException, NotPrimeException, EOFException
```

that reads a line of input from the keyboard and returns a value of type **int** if the string read represents a prime number. The method generates a `NotIntegerException` when the input string does not represent a valid integer and a `NotPrimeException` when the input string represents an integer that is not a prime. An `EOFException` signifies that the end-of-file character (Control-D in Unix) has been typed.

Write a loop in Java to read values from the keyboard using the method `System.in.readPrime()` and print the sum of the first 200 primes read. All lines with invalid input (not an integer or not a prime) should be ignored. The number of primes in the input may be less than 200 and is not known in advance. The input is terminated by an end-of-file character. *(3 marks)*

2. In the readers and writers problem, multiple readers may read a database simultaneously so long as no writer writes, but only one writer at a time may write into the database. When a writer is active, no readers should be active.

Let us abstractly represent the database by a simple integer value stored in a Java class. Design the class with methods

```
public int read()
public void write(int n)
```

so that invocations of `read()` and `write(i)` by reader and writer threads are properly synchronized according to the requirements given above. *(7 marks)*

3. Let $\mathsf{I} := \lambda x \cdot x$, $\mathsf{K} := \lambda xy \cdot x$ and $\mathsf{S} := \lambda xyz \cdot xz(yz)$. Compute normal forms for the following lambda expressions. *(6 marks)*

(a) $(\lambda y \cdot yyy)\ (\mathsf{KI}(\mathsf{SS}))$.

(b) $(\lambda yz \cdot zy)\ ((\lambda x \cdot xxx)(\lambda x \cdot xxx))\ (\lambda w \cdot \mathsf{I})$.

(c) SKK.

4. Recall that the Fibonacci numbers are given by the following recurrence: *(4 marks)*

$$
\begin{aligned}
F_0 &:= 0 \\
F_1 &:= 1 \\
F_{n+2} &:= F_n + F_{n+1}
\end{aligned}
$$

Define a lambda expression **fib** such that for all $m \in \mathbb{N}$, **fib** $\langle m \rangle \xrightarrow{*}_\beta \langle F_m \rangle$. (**Hint:** Use **pair**, **fst** and **snd** to define an appropriate step function from pairs to pairs, and iterate it on an appropriate initial pair.)

5. Define a lambda expression **odd** such that for all $m \in \mathbb{N}$, *(4 marks)*

odd $\langle m \rangle \xrightarrow{*}_\beta$ **true** if m is odd **odd** $\langle m \rangle \xrightarrow{*}_\beta$ **false** if m is even

6. Define a lambda expression **leq** such that for all $m, n \in \mathbb{N}$, *(4 marks)*

leq $\langle m \rangle \langle n \rangle \xrightarrow{*}_\beta$ **true** if $m \leqslant n$ **leq** $\langle m \rangle \langle n \rangle \xrightarrow{*}_\beta$ **false** if $m > n$

7. Define a lambda expression **isPOT** such that for all $m \in \mathbb{N}$, Assume half $(m) \xrightarrow{*}_\beta \langle \lfloor m/2 \rfloor \rangle$ *(4 marks)*

isPOT $\langle m \rangle \xrightarrow{*}_\beta$ **true** if m is a power of 2 **isPOT** $\langle m \rangle \xrightarrow{*}_\beta$ **false** if m is not a power of 2

Hint: Give an appropriate recursive equation that is satisfied by any power of 2, and use Θ to find a solution for the equation. No need to show how Θ is defined – just use the fact that $\Theta\, M \xrightarrow{*}_\beta M\, (\Theta\, M)$ for any M.

8. Assume that we have a constant **fix** of type $(\sigma \to \sigma) \to \sigma$ (for any type σ) and the following reduction rule:

$$\textbf{fix}\, F \xrightarrow{*}_\beta F\, (\textbf{fix}\, F).$$

Assume that we have a base types **int** and **bool**, and the following familiar constants and functions with the following types and the usual reduction rules:

⟨0⟩, ⟨1⟩, . . .	::	**int**	
pred	::	**int** → **int**	
mult	::	**int** → **int** → **int**	
iszero	::	**int** → **bool**	
ite	::	**bool** → σ → σ → σ	for any type σ

What is the type of the following expression F?

$$F \;:=\; \lambda f n \cdot \{\, \textbf{ite}\, (\textbf{iszero}\, n)\, \langle 1 \rangle\, (\textbf{mult}\, n\, (f\, (\textbf{pred}\, n))) \,\}.$$

If we define **fact** := **fix** F, show how **fact** ⟨3⟩ reduces to its normal form. *(8 marks)*