Indian Institute of Technology Delhi
Department of Computer Science and Engineering

COL226	Programming Languages	Minor 1 Test
February 6, 2020	60 minutes	Maximum Marks: 60

Q1	Q2	Q3	Q4	Q5	*Total*
10	16	10	14	10	60

Open notes. Write your name, entry number and group at the top of each sheet in the blanks provided. Answer all questions in the space provided, in blue or black ink (no pencils, no red pens). Budget your time according to the marks. Do rough work on separate sheets.

Q1. (10 marks) **Cross-compilation.** Recall that a *compiler* translates a program written in a *source language* X to a *target language* Y, and that an interpreter is an *executable* program that takes as input a program in source language X, and runs it on a specified (second) input. Also recall that one can only run a program that is in executable form (*i.e.*, in machine code and not in source language format) of the host machine. *Byte code* is the code that is interpreted on a *virtual machine*, whereas *native code* is machine code for a physical machine, and *source code* is program code written in (usually) a high-level programming language.

Suppose I have an *Andarm* machine and a *Lintel* machine, as well as the following software objects (copied onto both machines):

(a) File *sc.s* containing the source code, written in language *S*, of a compiler *sc*; when executed, *sc* will take programs written in language *S*, and produces *native code* for an *Andarm*-machine;

(b) File *sbc.b* containing a compiler *sbc* already compiled into *byte-code* language *B*; when executed, *sbc* takes programs in language *S* and produces equivalent code in byte-code *B*;

(c) An executable byte-code-*B*-interpreter bi_{Lintel} that runs on a *Lintel*-machine.

(d) File *foo.s* containing an *S*-language program and its input in file *inp*.

Indicate the steps by which (following a small variant on boot-strapping) I can produce a *native code compiler* for language *S*, which runs on *Andarm*-machines (and produces native *Andarm*-machine code), and use it to run the program in *foo.s* on input in *inp*:

	On machine	run program file	with input file	(and second input file)	yielding output file	
1	Lintel	bi_{Lintel}	sbc.b	sc.s	sc.b	3
2	Lintel	bi_{Lintel}	sc.b	sc.s	sc.a	3
3	Andarm	sc.a	foo.s		foo.a	2
4	Andarm	foo.a	inp			2
5						

Q2. (2+8+6=16 marks) **Type-checking.** Consider the following OCaml programs

(a) `let k x y = x;;`

What is the type of the function `k`? $\alpha \to (\beta \to \alpha)$

Working:

$k : \gamma$

$x : \delta$

$y : \theta$

$rhs : \pi$

$\therefore \boxed{\gamma = \delta \to (\theta \to \pi)}$ 1

$rhs = x \therefore \boxed{\delta = \pi}$ 1

renaming $\delta \mapsto \alpha$

$\theta \to \beta$

(b) `let s x y z = (x z) (y z);;`

What is the type of the function s? $\alpha \to (\beta \to \gamma) \to (\alpha \to \beta) \to (\alpha \to \gamma)$

Working:

$s : \delta$
$x : \theta$
$y : \pi$
$z : \sigma$
rhs : τ

$\therefore \boxed{\delta = \theta \to (\pi \to (\sigma \to \tau))}$ 1

Let $(x\ z) : \rho$ $\therefore \boxed{\theta = \sigma \to \rho}$ 2

Let $(y\ z) : \kappa$ $\therefore \boxed{\pi = \sigma \to \kappa}$ 2

$(x\ z)\ (y\ z) : \tau$ $\therefore \boxed{\rho = \kappa \to \tau}$ ~~2~~

$\therefore \boxed{\delta = (\underbrace{\sigma}_{\alpha} \to (\underbrace{\kappa}_{\beta} \to \underbrace{\tau}_{\gamma})) \to (\underbrace{\sigma}_{\alpha} \to \underbrace{\kappa}_{\beta}) \to (\underbrace{\sigma}_{\alpha} \to \underbrace{\tau}_{\gamma})}$ 1

(c) `let i = (s k) k ;;`

What is the type of the value i? $\delta \to \delta$

Working:

$s : (\alpha \to (\beta \to \gamma)) \to (\alpha \to \beta) \to (\alpha \to \gamma).$ } 1
$k : \delta \to (\theta \to \delta)$

For $(s\ k)$ to be well typed. $\boxed{\alpha = \delta}$ $\boxed{\theta = \beta}$ $\boxed{\delta = \gamma}$ 2

$\therefore (s\ k) : (\delta \to \theta) \to (\delta \to \delta)$ $k : \pi \to (\rho \to \pi)$

For $(s\ k)\ k$ to be well typed. $\boxed{\delta = \pi}$ $\boxed{\theta = \rho \to \pi}$ 2

$\therefore \boxed{(s\ k)\ k : \delta \to \delta}$ 1

Q3. (4+6=10 marks) **Interpreter.** Consider a language of boolean propositions. The abstract syntax may be coded in OCaml as follows:

```
type prop =  T | F  | Not of prop | And of prop * prop | Or of prop * prop;;
```

(a) Define a function size in OCaml that given an expressions of type prop returns the number of logical symbols (corresponding to each constructors) in it.

```
(* size:   prop -> int *)
let rec size p = match p with
    T -> 1
  | F -> 1
  | Not p1 -> 1 + (size p1)
  | And(p1, p2) -> 1 + (size p1) + (size p2)
  | Or(p1, p2) -> 1 + (size p1) + (size p2)
```
} 1
1
1
1

(b) Define the "mathematical" propositional interpreter function, which returns the boolean value of a given proposition. Write this as an OCaml function *eval* where expressions of type `prop` are mapped to values in the OCaml type `bool`:

```
(* eval:  prop -> bool *)
let rec eval p = match p with
    T -> true                                    —— 1
  | F -> false                                   —— 1
  | Not p1 -> not (eval p1)                      —— 1
  | And(p1, p2) -> (eval p1) && (eval p2)        —— 1.5
  | Or(p1, p2) -> (eval p1) || (eval p2)         —— 1.5
;;
```

Q4. (6+8=14 marks) **Functions and Relations.** Recall that a relation $R \subseteq A \times A$ is transitive if whenever $(a, b) \in R$ and $(b, c) \in R$, then $(a, c) \in R$. Also recall that an equivalence relation is reflexive, symmetric and transitive.

(a) Show that $R \subseteq A \times A$ is transitive if and only if $R \circ R \subseteq R$.

1 ⇒ 2 Suppose R is transitive 1 To prove $R \circ R \subseteq R$. 2

If $(a,b) \in R$ and $(b,c) \in R$, then $(a,c) \in R$ (Def of 1)

$\therefore$ $(a,c) \in R \circ R$ by defn of relation composition.

But a,b,c arbitrarily chosen. So $R \circ R \subseteq R$. 3

1 ⇐ 2 Suppose $R \circ R \subseteq R$.

~~Let~~ If $(a,c) \in R \circ R$. ~~$(a,c) \in R$.~~

$\therefore$ by defn of relation composition, for some $b \in A$ $(a,b) \in R$ and $(b,c) \in R$.

~~So whenever~~ Suppose $(a,b) \in R$, $(b,c) \in R$. 3

$\therefore$ $(a,c) \in R \circ R \subseteq R$

$\therefore$ $(a,c) \in R$.

(b) Let $f : A \to B$ be a total function. Define $R_f = \{(a_1, a_2) \in A \times A \mid f(a_1) = f(a_2)\}$. Show that R_f is an equivalence relation.

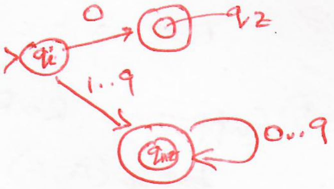
Reflexivity. for all $a \in A$ $f(a) = f(a)$ (f is total $\Rightarrow$ = is reflexive)

$\therefore$ $(a,a) \in R_f$. 2

Symmetry. If $(a,b) \in R_f$ then $f(a) = f(b)$.
But then $f(b) = f(a)$ (Symmetry of $=$) 3
$\therefore (b,a) \in R_f$.

Transitivity. If $(a,b) \in R_f$ and $(b,c) \in R_f$
then $f(a) = f(b)$ and $f(b) = f(c)$
$\therefore f(a) = f(c)$ (Transitivity of $=$)
3
$\therefore (a,c) \in R_f$.

Q5. (10 marks) **Grammars and FSMs.** Let $\mathcal{M} = (\mathcal{A}, \mathcal{Q}, \delta, q_0, \mathcal{F})$ be a Finite State Machine (FSM), where $\mathcal{A}$ is the alphabet, $\mathcal{Q}$ the set of states, $\delta \subseteq \mathcal{Q} \times \mathcal{A} \times \mathcal{Q}$ the transition relation (edges in the FSM), $q_0 \in \mathcal{Q}$ the start state, and $\mathcal{F} \subseteq \mathcal{Q}$ the set of accepting or final states. Any FSM can be coded as a Context-Free Grammar $\mathcal{G} = (\mathcal{N}, \mathcal{A}, \mathcal{P}, Q_0)$ as follows:



- Let the alphabet $\mathcal{A}$ be the set of terminals;
- With each state $q_i \in \mathcal{Q}$, associate a distinct non-terminal Q_i in $\mathcal{N}$;
- For each transition $(q_i, a, q_j) \in \delta$, introduce a production "$Q_i \longrightarrow aQ_j$" in $\mathcal{P}$;
- For each $q_j \in \mathcal{F}$, introduce a production rule "$Q_j \longrightarrow \epsilon$" in $\mathcal{P}$;
- Let Q_o be the start symbol.

Write a context-free grammar for *natural numerals* with no useless leading zeroes (e.g., 0 and 140 but not 007);

$\mathcal{N} =$ (Q_i, Q_z, Q_{nz})

$\mathcal{A} =$ $0 | 1 | \ldots | 9$

Start symbol Q_o is Q_i

Production rules $\mathcal{P}$ are:

$Q_i \to 0\, Q_z$

$Q_i \to 1 Q_{nz} | \ldots | 9\, Q_{nz}$

$Q_z \to \epsilon$

$Q_{nz} \to \epsilon$

$Q_{nz} \to 0 Q_{nz} | \ldots | 9\, Q_{nz}$