Indian Institute of Technology Delhi
Department of Computer Science and Engineering

CS 232 F | Programming Languages | Minor II
March 26, 2003 | 14:00–15:00 | Maximum Marks: 60

Answer all questions in the space provided, in blue or black ink (no pencils, no red pens). Budget your time according to the marks. Use the bottom of the last page for rough work. Unless otherwise stated, assume in the following that

$$c \in \mathit{Com} ::= \mathbf{skip} \parallel x := e \parallel c_1;\, c_2 \parallel \mathbf{if}\ e\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2\ \mathbf{fi} \parallel \mathbf{while}\ e\ \mathbf{do}\ c\ \mathbf{od}$$

$\mathcal{X}$ is a set of variable names, with x a typical variable. *Exp* is the inductively defined set of expressions, with e, e' typical expressions. *Types* is the collection of types with τ a typical type. $\Gamma \in \mathcal{X} \rightarrow_{fin} \mathit{Types}$ is a typical type assignment. $\Gamma \vdash e : \tau$ is the typing relation "e has type τ under type assumptions Γ on the variables".

Q1 [8 marks] **Operational Semantics.** Consider the command **repeat** c **until** e **taeper** (where c is a command and e is a boolean expression), which can be described informally as "execute command c repeatedly, stopping when the condition e becomes *true*". Give Big-step rule(s) for this command

Q2 [8+8 marks] **Reasoning.** We say that two commands c_1 and c_2 are "operationally equivalent" (written $c_1 \approx c_2$) if for all tables γ and all stores σ, σ': $\gamma \vdash \langle c_1, \sigma\rangle \implies_c \sigma'$ if and only if $\gamma \vdash \langle c_2, \sigma\rangle \implies_c \sigma'$.

Show that

(i) **repeat** c **until** b **taeper** $\approx$ c; **while not**(b) **do** c **od**
(ii) **while** b **do** c **od** $\approx$ **if** b **then** c; **while** b **do** c **od else skip fi**

Q3 [12 marks] **Type checking.** Strongly typed languages like Pascal are based on the notion of variables having types: if a variable x has type τ it means that x names a location that can contain values of type τ. Assuming that we are given the types of all the variables in a command, the command is well-formed if it does not contain "type errors", such as variables being assigned values of incorrect types, or the test expressions in "if" and "while" statements not being boolean. Complete the following inductive definition of a relation $\Gamma \Vdash c \surd$ that characterizes c being well-formed under type assumptions Γ.

$$\frac{}{\Gamma \Vdash \textbf{skip} \surd} \qquad \frac{\Gamma \Vdash c_1 \surd}{\Gamma \Vdash c_1;\, c_2 \surd}$$

$$\frac{}{\Gamma \Vdash x := e \surd} \qquad \frac{\Gamma \vdash e : \textbf{boolean}}{\Gamma \Vdash \textbf{while } e \textbf{ do } c \textbf{ od} \surd}$$

$$\frac{\Gamma \Vdash c_1 \surd}{\Gamma \Vdash \textbf{if } e \textbf{ then } c_1 \textbf{ else } c_2 \textbf{ fi} \surd}$$

Q4 [8+8+6+2 marks] **Principles and Blocks.** In most languages, when local variables are declared, we specify their types. Also it is considered good practice to initialize variables on declaration. Accordingly let the syntax of blocks be modified to

$$c ::= \ldots \mid \textbf{var } vd \textbf{ begin } c \textbf{ end} \qquad\qquad vd ::= x : \tau := e \parallel vd_1;\, vd_2$$

(a) Extend the rules for $\Gamma \Vdash c \surd$ for these modified construct(s).

(b) Provide big-step operational rules to specify the execution of the new constructs.

(c) According to the Principle of Correspondence, what code should be equivalent to the block **var** $x : \tau := e$ **begin** c **end**? (use a Pascal like syntax)?

(d) Suppose in Pascal that definitions of procedures were considered to be in parallel rather than in sequence (thus simplifying the answer somewhat), characterize in one sentence what procedures can be called by a nested procedure (no answers by examples, please!).

Take home portion: Implement **in SML** the type-checking and execution rules for the commands and blocks dealt with in this exam (note, it involves implementing the type-checking and evaluation rules for expressions as well).