Final Exam (COSE 312 Compilers, Spring 2025)

Student Number: **Name:**

Leave a blank when uncertain. Each correct answer gets you 1 point but you lose 1 point for each wrong answer.

(1) Big-step operational semantics for **While** is compositional. X

(2) The following function F has infinitely many fixed points. O

$$F(g) = \lambda s. \begin{cases} g(s) & \text{if } s(x) \neq 0 \\ s & \text{if } s(x) = 0 \end{cases}$$

(3) Let w be the following loop:

$$\texttt{while } \neg(x = 0) \texttt{ do } x := x - 1$$

Then,

$$C[\![\ w\ ]\!](s) = \begin{cases} s[x \mapsto 0] & \text{if } s(x) \geq 0 \\ \textbf{undef} & \text{if } s(x) < 0 \end{cases}$$

O

(4) $\emptyset$ is a chain. O

(5) Every CPO has the least element, denoted $\bot$. O

(6) Every CPO has the greatest element, denoted $\top$. X

(7) The following function is continuous:

$$F_1(g) = \begin{cases} g_1 & \cdots g = g_2 \\ g_2 & \cdots \textit{otherwise} \end{cases}$$

where $g_1 \neq g_2$. X

(8) The semantic function $C[\![\ c\ ]\!] : \textbf{State} \hookrightarrow \textbf{State}$ for commands is strict, i.e., $C[\![\ c\ ]\!](\bot) = \bot$. O

(9) $C[\![\ c\ ]\!]$ is a partial function. O

(10) $\widehat{C}[\![\ c\ ]\!]$ is a partial function. X

(11) Sound static analysis can prove the presence of bugs. X

(12) The simple sign analysis discussed in class is guaranteed to terminate even for non-terminating programs. O

(13) The simple sign domain, $\{\top, \bot, \text{neg}, \text{zero}, \text{pos}\}$, is a complete lattice. O

(14) The interval domain is not a complete lattice. X

(15) For some programs, interval analysis terminates without the use of widening. O

(16) For some programs, concrete execution terminates but interval analysis without widening does not terminate. O

(17) Widening and narrowing ensure termination with the least fixed point. X

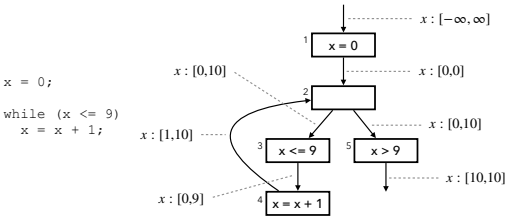
(18) Fixed point computation with narrowing follows a decreasing chain. O

(19) $[1,2] =_{\widehat{\mathbb{Z}}} [1,2]$. X

(20) The interval domain is a relational domain. X

(21) For finite height domains with infinite elements, fixed point computation always terminates. O

(22) The following analysis result represents the least fixed point; any result more precise than this would be unsound. O



(23) Suppose programs P_1 and P_2 are semantically equivalent. Then, interval analysis is guaranteed to produce identical results for both programs. X

(24) Consider the assignment $lv := e$, where assume the input abstract state is $\hat{m}$ and $\widehat{\mathcal{L}}(lv)(\hat{m}) = \{x, y\}$. Then, the abstract semantics that produces the following output state

$$\hat{m}[x \mapsto \widehat{\mathcal{E}}(e)(\hat{m}),\ y \mapsto \widehat{\mathcal{E}}(e)(\hat{m})]$$

is sound. X

(25) It is sound to perform loop-invariant code motion when the reaching definition information is over-approximated. O

(26) Reaching definition analysis is a forward, must analysis. X

(27) Liveness analysis is a backward, may analysis. O

(28) Exact def/use sets are not computable for Turing-complete languages. O

(29) The available expressions analysis computes only the truly available expressions. O

(30) In a must data-flow analysis, the join operator ($\sqcup$) corresponds to the set union ($\cup$). x

(31) In a must data-flow analysis, the greatest fixed point corresponds to the most precise solution. O

(32) A must data-flow analysis computes an increasing chain. X

(33) Suppose the constant propagation analysis has computed the abstract state:

$$\{x \mapsto \top, y \mapsto 3, z \mapsto \top\}$$

at a program point. This means that y is guaranteed to be constant 3 at that point no matter how the program is executed. O

(34) The "busy" expression analysis mentioned in class is a must, backward analysis. O

(35) Interval analysis is always at least as precise as constant propagation analysis. O

(36) A may analysis (e.g., constant propagation) cannot be used to infer "must" program properties. X