Consider the tiny imperative language on mathematical integers (similar to that of class 8),

\[ V ::= x \mid y \mid \ldots \] variables
\[ E ::= 0 \mid 1 \mid V \mid E - E \mid \text{input()} \mid \text{sanitize}(V) \] expressions
\[ C ::= \text{skip} \mid \text{if } E \text{ then } C \text{ else } C \text{ fi} \mid \text{while } E \text{ do } C \text{ od} \] commands

where the test is false for 0 and true for all other integers.

To deal with security issues, several programming languages have a notion of “tainted” data. The idea is that any value read from the outside environment by \text{input}() is marked as being tainted, \textit{i.e.} potentially dangerous/unsafe, whereas \text{sanitize}(V) returns a cleaned value of \( V \) which is assumed to be no-longer tainted (\textit{e.g.} because of a sanity check). The result of any assignment that uses tainted data are also marked tainted.

Consider the complete lattice \( L = \{ \bot, \top \} \) such that \( \bot \sqsubseteq \bot \sqsubseteq \top \sqsubseteq \top \) where \( \bot \) means “definitely not tainted” and \( \top \) means “potentially tainted”. The purpose of the exam is to design a static tainting analysis for the above tiny imperative language. We let \( R = V \rightarrow L \) be the set of environments \( \rho \in R \) providing the tainting information \( \rho(x) \in L \) of each program variable \( x \in V \). Initially \( \forall x \in V : \rho(x) = \bot \).

**Question 1** Consider the program

\[ y := 1; \text{while } y<0 \text{ do } x := \text{input}(); y := y + \text{sanitize}(x) \text{ od} \]

Is the final value of \( y \) clean or tainted? \( \square \)

**Answer 1** \( y \) depends only upon the initial value 1 which is clean and on the sanitized value of \( x \) so it is not tainted. \( \square \)

**Question 2** Define the join \( \sqcup \) and the meet \( \sqcap \) in the lattice \( L \).

**Answer 2** \( \bot \sqcup \bot = \bot, \top \sqcup x = \top, x \sqcap x = \bot, x \sqcap \bot = \bot \sqcap x = \bot. \square \)
Question 3 Define the infimum, join $\sqcup$, and the meet $\sqcap$ in the lattice $R$ for the pointwise ordering $\rho \sqsubseteq \rho'$ if and only if $\forall x \in V : \rho(x) \sqsubseteq \rho'(x)$. □

Answer 3 The infimum is $\lambda x \cdot \bot$, the join $\rho \sqcup \rho' = \lambda x \cdot \rho(x) \sqcup \rho'(x)$ and the meet $\rho \sqcap \rho' = \lambda x \cdot \rho(x) \sqcap \rho'(x)$. □

Question 4 For all expressions $E$, define $T\llbracket E \rrbracket \in R \rightarrow L$ such that if $\rho \in R$ specifies which variables are tainted then $T\llbracket E \rrbracket \rho$ specifies whether the value of $E$ is tainted i.e. because it uses one variable which value is tainted. □

Answer 4

\[
\begin{align*}
T\llbracket 0 \rrbracket \rho &= T\llbracket 1 \rrbracket \rho = \bot \\
T\llbracket V \rrbracket \rho &= \rho(V) \\
T\llbracket E_1 - E_2 \rrbracket \rho &= T\llbracket E_1 \rrbracket \rho \sqcup T\llbracket E_2 \rrbracket \rho \\
T\llbracket E \rrbracket \rho &= T\llbracket E \rrbracket \rho \\
T\llbracket \text{input}() \rrbracket \rho &= \top \\
T\llbracket \text{sanitize}(V) \rrbracket \rho &= \rho(V) \sqcap \bot = \bot
\end{align*}
\]

Question 5 For $C = \text{skip}$, $V := E$, $C_1 ; C_2$, or if $E$ then $C_1$ else $C_2$ fi define $T\llbracket C \rrbracket \rho$ specifying which variables are tainted after executing command $C$ assuming that $\rho \in R$ specifies which variables are tainted before executing $C$. □

Answer 5

\[
\begin{align*}
T\llbracket \text{skip} \rrbracket \rho &= T\rho \\
T\llbracket V := E \rrbracket \rho &= \rho[V := T\llbracket E \rrbracket] \\
T\llbracket C_1 ; C_2 \rrbracket \rho &= T\llbracket C_2 \rrbracket (T\llbracket C_1 \rrbracket \rho) \\
T\llbracket \text{if } E \text{ then } C_1 \text{ else } C_2 \text{ fi} \rrbracket \rho &= T\llbracket C_1 \rrbracket \sqcup T\llbracket C_2 \rrbracket.
\end{align*}
\]

where $\rho[V := t](V) = t$ and $\rho[V := t](V') = \rho(V')$ when $V' \neq V$. □

Question 6 Define $T\llbracket \text{while } E \text{ do } C \text{ od} \rrbracket \rho$ specifying which variables are tainted after executing the iteration command while $E$ do $C$ od assuming that $\rho \in R$ specifies which variables are tainted before executing the loop. □

Answer 6

\[
T\llbracket \text{while } E \text{ do } C \text{ od} \rrbracket \rho = \bigsqcup_{n \geq 0} T\llbracket C \rrbracket^n(\rho)
\]
i.e. $\rho \sqcup T\llbracket C \rrbracket \rho \sqcup T\llbracket C \rrbracket (T\llbracket C \rrbracket \rho) \sqcup \ldots$ after 0, 1, 2 or more iterations in the loop (which is also $\text{ifp} \equiv F[C] \rho$ where $F[C] \rho = \rho \sqcup T\llbracket C \rrbracket X$). □

Question 7 Prove that it is undecidable to determine whether a program variable is definitely tainted. □

Answer 7 Assume by reductio ad absurdum that tainting is decidable that is can be answered statically (at compile time). Then termination of program $P$ would be decidable by considering the following program $Q$ (where $x$ and $y$ are fresh integer variables not in $P$)
input(y);
P;
x:=y.

since x is tainted if and only if P terminates, a contradiction.

**Question 8** Provide an example of a very simple program with a variable which is not tainted by the analysis can’t determine that fact.

**Answer 8** input(y); x = y - y
x is 0 so does not depend upon the input y so is not tainted.