1. Problem 1 (30 Points)
In the PowerPoint of Lecture 27, slide 21, at the end of the definition of the Pascal "for" loop, why do we have "(Replace m I \bottom, i, o)"?

Solution:
Because according to Pascal specification, the loop variable has to be undefined after the execution of the loop; its value is reset to undefined.

2. Problem 2 (30 Points)
In the same Powerpoint lecture, slide 23, at the end of the definition of the Pascal "case" statement, why do we have "o (\lambda(v,s).s)"?

Solution:
The goal is to discard the value because a statement should return a state not a value-state tuple.

3. Problem 3 (40 Points)
Add a 'do' statement (as in the C language) to the denotational semantics specification of Tiny.
Solution:
First, add an item to the C syntactic domain:
\[
C = \ldots \mid \langle\text{do } C \ E\rangle
\]
Then add a case in the definition of CC. The trick is to notice that
\[
do C \text{ while } E
\]
is equivalent to
\[
C;
\]
\[
\text{while } E \text{ do } C;
\]
So,
\[
\text{CC}[\langle\text{do } C \ E\rangle] = \text{CC}[<; C <\text{while } E C>>]
\]

4. Problem 4 (30 Points)
Consider the denotational semantics specification of Tiny. Suppose we were to make a single change to the specification of the ‘+’ operator (slide 20 lecture ppt 26). Consider changing the LAST occurrence of s2 to s1, on the next-to-last line. Explain in plain English how the semantics of the Tiny language change due to this change in the specification. Illustrate your argument by constructing a sample Tiny program that illustrates the different semantics.

Solution:
program x:
\[
\begin{align*}
\text{assign } i & := 1 + \text{read}; \\
\text{output}(\text{read}); \\
\end{align*}
\]
end x.

5. Problem 5 (30 Points)
Why is type checking difficult in the denotational description of Tiny?

Solution:
Because there is no variable declaration in Tiny.
6. Problem 6 (40 Points)

Write the denotational description of the 'case' statement in C for Tiny.

Solution:

\[
C_{CC}[<c\_c n C>] = \lambda(v,s). \begin{cases} 
  v eq \text{\texttt{undef}} \rightarrow (v,s) \\
  v ne n \rightarrow (v,s) \\
  (\text{\texttt{undef}}, s \Rightarrow CC[C])
\end{cases}
\]