Semantics of Programming Languages

The language L has expression syntax

\[ e ::= n | \textbf{skip} | \textbf{fn} \, x : T \Rightarrow e | e_1 \, e_2 | x | e_1 ::= e_2 | !e | \textbf{ref} \, e | \ell \]

with types

\[ T ::= \text{int} | \text{unit} | T_1 \rightarrow T_2 | T \, \text{ref} \]

It is intended to have a call-by-value semantics.

\( (a) \) Define the set of values for L. \[2 \text{ marks}\]

\( (b) \) Give type rules and reduction rules for the store-related expressions \[ e_1 ::= e_2 | !e | \textbf{ref} \, e | \ell. \]
Define clearly what the type environments and stores you are using are. \[10 \text{ marks}\]

\( (c) \) Discuss possible alternative choices for the semantics of the operations in part \( (b) \), paying particular attention to: (i) what can be stored, (ii) store initialisation, and (iii) the results of assignments. Illustrate your answer with type rules and reduction rules as appropriate, and comment on any pragmatic advantages or disadvantages. \[8 \text{ marks}\]