Semantics of Programming Languages

Consider the following syntax for a pure untyped functional language.

Booleans \( b \in \mathbb{B} = \{ \text{true}, \text{false} \} \)

Integers \( n \in \mathbb{Z} = \{ ..., -1, 0, 1, ... \} \)

Variables \( x \in \mathbb{X} \) for a set \( \mathbb{X} = \{ x, y, z, ... \} \)

Operations \( \text{op ::= + | \geq } \)

Expressions \( e ::= \text{skip} | n | b | e_1 \text{ op } e_2 | \text{if } e_1 \text{ then } e_2 \text{ else } e_3 | \text{fn } x \Rightarrow e | e_1 \text{ e}_2 | x | \text{fix } e \)

The language supports recursion with a fixed-point operator \( \text{fix } e \), which has semantics defined by the rule below.

\[
\overline{\text{fix } e} \rightarrow e(\overline{\text{fix } e})
\]

(a) Give the semantic rules for function application for call-by-value, call-by-name, and full-beta reduction for this language (do not give the rules for binary operators, conditional, or fix). You should define a small-step reduction relation \( e \rightarrow e' \), stating precisely what notion of values \( v \) you are using. [10 marks]

(b) For the call-by-value semantics, characterise the expressions \( e \) from the grammar above that have an immediate runtime error in their outermost (top-level) construct. [3 marks]

(c) For each pair of semantics (call-by-value and call-by-name, call-by-name and full-beta, and full-beta and call-by-value), give an expression with different possible termination behaviours in each element of the pair. [4 marks]

(d) For each of your three semantics, explain a disadvantage in using that semantics for a programming language. [3 marks]