Semantics of Programming Languages

Let L be the language with syntax below, a call-by-value left-to-right operational semantics, and the standard simple type system.

\[
\begin{align*}
    e & ::= \text{n} \mid \text{fn}\ x:T \Rightarrow e \mid e_1\ e_2 \mid x \\
    T & ::= \text{int} \mid T_1 \rightarrow T_2 \\
    n & \in \mathbb{Z}
\end{align*}
\]

(a) L is not Turing-complete: there are computable functions over the integers that are not expressible as closed L expressions of type int → int. Why not? \[2 \text{ marks}\]

(b) Define a modest extension L’ of L that is Turing-complete. Give the additional syntactic forms (for expressions and types), describe their operational semantics informally, and state their precise typing rules. \[10 \text{ marks}\]

(c) Assuming that f is an L’ expression of type int → int, give an expression e of type int that computes the smallest x such that f x is zero. Explain how this can be used to prove completeness. \[4 \text{ marks}\]

(d) Discuss whether Turing-completeness is a necessary or sufficient property for a good programming language. \[4 \text{ marks}\]