Semantics of Programming Languages

Consider the variant of untyped L1 with syntax as below and a standard small-step semantics $\langle e, s\rangle \longrightarrow \langle e', s'\rangle$ (this is identical to L1 except that it has equality testing $e_1 = e_2$ on integers instead of $\geq$ and that here stores are total functions).

Booleans $b \in \mathbb{B} = \{\textsf{true}, \textsf{false}\}$
Integers $n \in \mathbb{Z} = \{..., -1, 0, 1, ...\}$
Locations $\ell \in \mathbb{L} = \{l, l_0, l_1, l_2, ...\}$
Stores s, total functions from $\mathbb{L}$ to $\mathbb{Z}$
Values $v ::= \textsf{skip} \mid n \mid b$
Operations $op ::= \ = \mid +$
Expressions

$$e ::= \textsf{skip} \mid n \mid b \mid e_1 \ op \ e_2 \mid \textsf{if}\ e_1\ \textsf{then}\ e_2\ \textsf{else}\ e_3 \mid \ell := e \mid !\ell \mid e_1; e_2 \mid \textsf{while}\ e_1\ \textsf{do}\ e_2$$

Define $[\![e]\!]$ to be the function that takes any store s and either is $\bot$ (undefined), if $\langle e, s\rangle \longrightarrow^{\omega}$, or is $\langle v, s'\rangle$, if $\langle e, s\rangle \longrightarrow^{*} \langle v, s'\rangle$.

Define (untyped) semantic equivalence $e_1 \simeq e_2$ iff $[\![e_1]\!] = [\![e_2]\!]$.

(*a*) State what it means for $\simeq$ to be a congruence. [2 marks]

(*b*) For each of the constructs of the expression grammar, define an explicit characterisation of $[\![e]\!]$ in terms only of the semantics $[\![e']\!]$ of its subexpressions e', without using the reduction relation. (For example, for n (which has no subexpressions) $[\![n]\!] = \lambda s.\langle n, s\rangle$.) [12 marks]

(*c*) Consider $(\textsf{if}\ !l = 1\ \textsf{then}\ e\ \textsf{else}\ e) \simeq e$. Either prove it, using your answer to part (*b*), or exhibit a counterexample. [3 marks]

(*d*) Consider $(\textsf{while}\ e_1\ \textsf{do}\ e_2) \simeq (\textsf{while}\ e_1\ \textsf{do}\ (e_2; e_2))$ where e_1 does not read any store locations. State whether this is true or false, with an informal explanation of the possible cases. [3 marks]