Semantics of Programming Languages

The phrases, $P$, of the language LC are specified by:

\[
P :: \quad C \mid E \mid B
\]

\[
C :: \quad \text{skip} \mid \ell := E \mid C ; C \mid \text{if } B \text{ then } C \text{ else } C \mid \text{while } B \text{ do } C
\]

\[
E :: \quad n \mid \ell \mid E \ iop \ E
\]

\[
B :: \quad \text{true} \mid \text{false} \mid E \ bop \ E
\]

where $\ell$ ranges over storage locations, $n$ over integers, $iop$ over integer-valued operations, and $bop$ over boolean-valued operations. Describe the operational semantics of LC in terms of an inductively defined transition relation, $\rightarrow$, between configurations $\langle P, s \rangle$, where $s$ is a finite partial function from locations to integers. State which are the terminal configurations and explain what it means for a configuration to be stuck. \[6 \text{ marks}\]

Call a configuration $\langle P, s \rangle$ sensible if the set of locations on which $s$ is defined, $\text{dom}(s)$, contains all the locations that occur in the phrase $P$. Prove by induction on the structure of $P$ that for all $s$, if $\langle P, s \rangle$ is sensible then $\langle P, s \rangle$ is not stuck. \[6 \text{ marks}\]

Prove by Rule Induction for $\rightarrow$ that if $\langle P, s \rangle \rightarrow \langle P', s' \rangle$ and $\langle P, s \rangle$ is sensible, then so is $\langle P', s' \rangle$ and $\text{dom}(s') = \text{dom}(s)$. Deduce that a stuck configuration can never be reached by a series of transitions from a sensible configuration. \[8 \text{ marks}\]