The syntax of commands of a simple imperative language with a repeat construct is given by:

\[ c ::= X := e \mid c_0 ; c_1 \mid \text{if } b \text{ then } c_0 \text{ else } c_1 \mid \text{repeat } c \text{ until } b \]

where \( X \) is a location, \( e \) is an arithmetic expression, \( b \) a Boolean expression and \( c, c_0, c_1 \) range over commands. A repeat statement executes statement \( c \) and then tests expression \( b \). If \( b \) is true, the loop terminates; otherwise, the loop continues.

1. Define an operational semantics in the form of rules to generate transitions of the form \( <c, \sigma> \rightarrow \sigma' \) meaning the execution of \( c \) from state \( \sigma \) terminates in state \( \sigma' \);

2. Define a denotational semantics for commands in which each command \( c \) is denoted by a partial function \( C[|c|] \) from states to states;

3. Sketch the proof of the equivalence between the operational and denotational semantics, that \( <c, \sigma> \rightarrow \sigma' \) iff \( C[|c|] \sigma = \sigma' \), only giving the case where \( c \) is a repeat loop.