7.1 We introduce symbols to represent variables whose value may change, but we do not bother to introduce symbols for variables whose value remains unchanged in the code we are symbolically executing. Why are new symbols necessary in the former case but not in the latter?

Symbolic execution builds predicates that characterize the conditions under which execution paths can be taken and the effect of the execution on program state. We do not introduce symbols for variables whose value remains unchanged because they will never have an effect on the program state. Symbols that do change can have constraints placed on them that could affect the outcome of the program. For example, for(i = 0; i < 10; i++) in symbolic execution would be traced through the branch in either direction. Execution of the test is adding a constraint to record the outcome.

7.5 Write the pre-and postconditions for a program that finds the index of the maximum element in a nonempty set of integers.

(| ∀i && 0 < val.size && 0 ≤ i < size && val.type = int |)
S = findMaxElement(val, val.size)
(| s = v && ∃i && 0 ≤ i < size : val[v] ≥ val[i] && returnVal = val[v] |)