**Slice-Based Testing Definitions:**

* Given a program P, and a program graph G(P) in which statements and statement fragments are numbered, and a set V of variables in P, the ***slice on the variable set V at statement fragment n***, written S(V,n), is the set node numbers of all statement fragments in P prior to n that contribute to the values of variables in V at statement fragment n
* The idea of slices is to separate a program into components that have some useful meaning
* We will include CONST declarations in slices
* Five forms of usage nodes
  + P-use (used in a predicate (decision))
  + C-use (used in computation)
  + O-use (used for output, e.g. writeln())
  + L-use (used for location, e.g. pointers)
  + I-use (iteration, e.g. internal counters)
* Two forms of definition nodes
  + I-def (defined by input, e.g. readln())
  + A-def (defined by assignment)
* For now, we presume that the slice S(V,n) is a slice on one variable, that is, the set V consists of a single variable, v
* If statement fragment n (in S(V,n)) is a defining node for v, then n is included in the slice
* If statement fragment n (in S(V,n)) is a usage node for v, then n is not included in the slice
* P-uses and C-uses of other variables are included to the extent that their execution affects the value of the variable v
* O-use, L-use, and I-use nodes are excluded from slices
* Consider making slices compliable