Ch8 1.

Binary search trees are relatively simple to implement. They are slower than hash tables. Insertion and lookup can be O(log­2 n) where n is the names in the tree. The average case is much slower than that at O(n) when a tree of n names has a depth of n.

Hash tables are more complicated to implement but faster than binary search trees. Insertion and lookup are O(1).

Ch9 2.

**class NodeVisitor**

**procedure** visitChildren(n)

**foreach** c ∈ n.getChildren( ) **do call** c.accept( **this** )

**end**

**end**

**class SemanticsVisito**r **extends NodeVisitor**

**procedure** checkInteger( c )

**if** c.type != Integer **and** c.type != errorType

**then call** error(”Require Integer type at”, c )

**end**

**procedure** visit(signTest signTestn)

**call** visitChildren( signTestn)

**call** checkInteger( signTestn.condition)

**end**

**end**

**class ReachabilityVisitor extends NodeVisitor**

**procedure** visit( signTesting signTestn)

signTestn.thenPart.isReachable← true

signTestn.elsePart.isReachable← true

**call** visitChildren(signTestn)

negNormal ←signTestn.negPart.terminatesNormally

zeroNormal ← signTestn.zeroPart.terminatesNormally

posNormal ← signTestn.posPart.terminatesNormally

signTestn.terminatesNormally← negNormal **or** zeroNormal **or** posNormal

**end**

**end**

**class ThrowsVisitor extends NodeVisitor**

**procedure gatherThrows(n)**

**call visitChildren(n)**

**ans ← ∅**

**foreach c ∈ n.getChildren( ) do ans ← ans ∪ c.throwsSet**

**n.throwsSet ← ans**

**end**

**procedure visit( signTesting signTestn)**

**call gatherThrows( i fn)**

**end**

**end**