```
treeVR( T, regList )
T is the root of an expression tree, regList is a
list of HardwareReg or VirtualReg objects.  |regList| must be
sufficiently sized for T evaluation.

This algorithm implements register targetting, the
result of T will be stored at head(regList).

// some utility list access definitions
head({l1,l2,l3,...}) → l1
tail({l1,l2,l3,...}) → {l2,l3,...}
head and tail do not modify their list argument.

r1 ← head(regList)
r2 ← head(tail(regList))

if ( T.kind is IntegerIdentifier ) then (
     r1.loadIntegerIdent(<ident>)
) else if ( T.kind is FloatIdentifier ) then (
     r1.loadFloatIdent(<ident>)
) else if ( T.kind is IntegerLiteral ) then (
     r1.loadIntegerImmediate(<value>)
) else if ( T.kind is FloatLiteral ) then (
     r1.loadFloatImmediate(<value>)
) else if ( ··· other one-register ops ) then (
     ⋮
) else if ( T.kind is BinaryOp ) then
   left ← T.leftTree
   right → T.rightTree
   // Eval the "larger" first
   if ( left.regCount > right.regCount ) then (
        treeCG( left, regList )      // result in r1 due to register targeting
        treeCG( right, tail(regList) )      // result in r2
        r1.binaryOp( <binaryOp>, r1, r1, r2 )
   ) else (
        treeCG( right, regList )
        treeCG( left, tail(regList) )
        r1.binaryOp(<binaryOp>,r1,r2,r1)
   )
   ) else if ( T.kind is FunctionCall ) then (
     functionCall( T, regList, ··· )    // Next lecture :)
   )
)
```