The ADT ‘Graph’

```
module Graph ( Graph, Node, Edge,
               emptyGraph, nodes, edges, s2n, n2s, ns2e, e2ns,
               insertNode, insertEdge, outEdges ) where

-------------------------------------------------------------------------------
-- I N T E R F A C E  :  P U B L I C
-------------------------------------------------------------------------------

-- Graph : an unweighted directed graph

-------------------------------------------------------------------------------

-- Node : a node

-------------------------------------------------------------------------------

-- Edge : an edge

-------------------------------------------------------------------------------

-- emptyGraph : the empty graph

emptyGraph :: Graph

-------------------------------------------------------------------------------

-- nodes g : a list of the nodes in graph ‘g’

nodes :: Graph -> [ Node ]

-------------------------------------------------------------------------------

-- edges g : a list of the edges in graph ‘g’

edges :: Graph -> [ Edge ]

-------------------------------------------------------------------------------

-- s2n s : a node with label ‘s’

s2n :: String -> Node

-------------------------------------------------------------------------------

-- n2s n : the label of node ‘n’

n2s :: Node -> String

-------------------------------------------------------------------------------

-- ns2e ( n1, n2 ) : an edge from node ‘n1’ to node ‘n2’

ns2e :: ( Node, Node ) -> Edge

-------------------------------------------------------------------------------

-- e2ns e : a tuple of the starting and finishing nodes of edge ‘e’

e2ns :: Edge -> ( Node, Node )

-------------------------------------------------------------------------------

-- insertNode n g : the graph formed by inserting node ‘n’ into graph ‘g’

insertNode :: Node -> Graph -> Graph

-------------------------------------------------------------------------------

-- insertEdge e g : the graph formed by inserting edge ‘e’ into graph ‘g’

insertEdge :: Edge -> Graph -> Graph

-------------------------------------------------------------------------------

-- outEdges n g : a list of the edges starting from node ‘n’ in graph ‘g’

outEdges :: Node -> Graph -> [ Edge ]

-------------------------------------------------------------------------------
```

The ADT ‘Graph’

```
-------------------------------------------------------------------------------
-- I M P L E M E N T A T I O N  :  P R I V A T E
-------------------------------------------------------------------------------

-- SIMPLIFIED IMPLEMENTATION : NO ERROR-CHECKING PERFORMED

-------------------------------------------------------------------------------

data Graph = G [ Node ] [ Edge ]

-------------------------------------------------------------------------------

data Node = N String deriving Eq

-------------------------------------------------------------------------------

data Edge = E Node Node deriving Eq  -- ( start, finish )

-------------------------------------------------------------------------------

emptyGraph = G [ ] [ ]

-------------------------------------------------------------------------------

nodes ( G ns _ ) = ns

-------------------------------------------------------------------------------

edges ( G _ es ) = es

-------------------------------------------------------------------------------

s2n s = N s

-------------------------------------------------------------------------------

n2s ( N s ) = s

-------------------------------------------------------------------------------

ns2e ( n1, n2 ) = E n1 n2

-------------------------------------------------------------------------------

e2ns ( E n1 n2 ) = ( n1, n2 )

-------------------------------------------------------------------------------

insertNode n ( G ns es ) = G ( n : ns ) es

-------------------------------------------------------------------------------

insertEdge e ( G ns es ) = G ns ( e : es )

-------------------------------------------------------------------------------

outEdges n ( G _ es ) = [ e | e <- es, fst ( e2ns e ) == n ]

-------------------------------------------------------------------------------
```