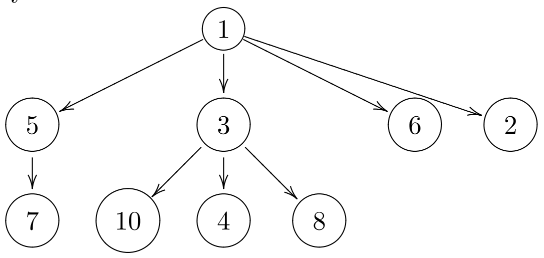
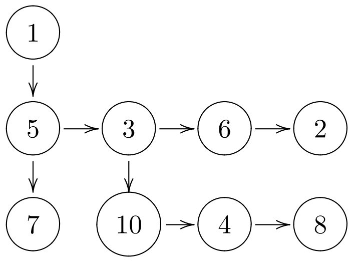
CEng242 Homework 1

Due 30th March 2004

A n-ary tree can be represented as a binary tree. In this representation left link will be a link to first child and right link will be a link to next sibling. In other words children information will be a linked list of trees.
Consider the following n-ary tree:



It can be represented as the binary tree:



Note that in this representation, root element (a standalone subtree as weell) has only one outgoing edge, the sibling edge is nil. For these two representations, you have following Haskell datatypes for integer trees:

```haskell
data NaryTree a = Node (a,[NaryTree a]) deriving Show
data NaryBin a = Bin (a, NaryBin a, NaryBin a) | Nil deriving Show
```

In NaryTree the children is kept as a list of NaryTree values. A leaf node will have empty list as the child information, like `Node(1,[])`. In NaryBin Bin (a,b,c) should be read 'a' as node content, 'b' as first child, 'c' as the next sibling.
Write the following two functions to convert trees between these representations:

binton : NaryBin $\alpha \longrightarrow$ NaryTree α

ntobin : NaryTree $\alpha \longrightarrow$ NaryBin α

Sample run:

```haskell
a = Node (1 ,[ Node (5 ,[ Node (7 ,[])]) ,
               Node (3 ,[ Node (10 ,[]) , Node (4 ,[]) , Node (8 ,[]) ]) ,
               Node (6 ,[]),
               Node (2 ,[])])

b=Bin(1 , Bin (5, Bin (7,Nil,Nil), Bin (3,Bin(10,Nil,
                                            Bin(4,Nil, Bin(8,Nil,Nil) )),
                                      Bin(6,Nil, Bin(2,Nil,Nil)))),
          Nil)
------------- Above in the code, following in the ghci prompt ------
ntobin a
Bin (1,Bin (5,Bin (7,Nil,Nil),Bin (3,Bin (10,Nil,Bin (4,Nil,Bin (8,Nil,Nil))),
   Bin (6,Nil,Bin (2,Nil,Nil)))),Nil)
-- above is the same value with b ---

binton b
Node (1,[Node (5,[Node (7,[])]),Node (3,[Node (10,[]),Node (4,[]),Node (8,[])]),
Node (6,[]),Node (2,[])])
--  above is the same value with a ---
```

Hint:
Please note that ntobin needs to return a root node which has no sibling (like (Bin (3, ..., Nil)).

Write a helper function that gets a list of child nodes and return a binary tree node with its possible siblings Note that mutually recursive calls are possible. Same idea can be applied to binton. It is always called with a node without a sibling. In a node with a sibling, the node isolated from the sibling and the list coming from the sibling can be combined into a list of NaryTree that will be the children of the value binton returns.
Examples:
Multi line inputs below should be entered on a single line on input.

```
t1 = Node (1,[])
t2 = Node (2, [Node (3, [Node (4, [])]),
              Node (1,[])])
t3 = Node (1, [t2, t1])
t4 = Node (9, [t1,t3,t2])
----- above part is in the code (not submitted)- below in ghci prompt---
Hw1> ntobin t1
Bin (1,Nil,Nil)

Hw1> ntobin t2
Bin (2,Bin (3,Bin (4,Nil,Nil),Bin (1,Nil,Nil)),Nil)

Hw1> ntobin t3
Bin (1,Bin (2,Bin (3,Bin (4,Nil,Nil),Bin (1,Nil,Nil)),Bin (1,Nil,Nil)),Nil)

Hw1> ntobin t4
Bin (9,Bin (1,Nil,Bin (1,Bin (2,Bin (3,Bin (4,Nil,Nil),Bin (1,Nil,Nil)),
                                Bin (1,Nil,Nil)),
                         Bin (2,Bin (3,Bin (4,Nil,Nil),Bin (1,Nil,Nil)),
                                Nil))),
       Nil)
-- the result above is indented to make it readable --

Hw1> binton (Bin (1,Nil,Nil))
Node (1,[])

Hw1> binton (Bin (2,Bin (3,Bin (4,Nil,Nil),Bin (1,Nil,Nil)),Nil))
Node (2,[Node (3,[Node (4,[])]),Node (1,[])])

Hw1> binton (Bin (9,Bin (1,Nil,Bin (1,Bin (2,Bin (3,
Bin (4,Nil,Nil),Bin (1,Nil,Nil)),Bin (1,Nil,Nil)),Bin (2,
Bin (3,Bin (4,Nil,Nil),Bin (1,Nil,Nil)),Nil))),Nil))

Node (9,[Node (1,[]),
         Node (1,[Node (2,[Node (3,[Node (4,[])]),
                           Node (1,[])]),
                  Node (1,[])]),
         Node (2,[Node (3,[Node (4,[])]),
                  Node (1,[])])])

-- the result above is indented to make it readable --
```

Two following functions are given to display trees in an easy to read format (development purpose only do not include in your submission!):

```
prettyb t =
   let pretty2 Nil _ _ ind = ind ++ "*"
       pretty2 (Bin (x, left , right)) first second ind =
                        let nfirst = first ++ "␣|␣␣␣"
                            nsecond = second ++ "␣|␣␣␣"
                            firstnull = first ++ "␣␣␣␣␣"
                            secondnull = second ++ "␣␣␣␣␣"
                            innf = first ++ "␣/---"
                            inns = second ++ "␣\\---"
                        in (pretty2 right firstnull nfirst innf) ++ "\n"
                            ++ ind ++ (show x) ++ "+\n"
                            ++ (pretty2 left nsecond secondnull inns)
   in putStr (pretty2 t "" "" "" ++"\n")

prettyn t =
   let pretty2 (Node (x,list)) ind nind = nind ++ "--␣" ++ (show x) ++ "\n" ++
                                 (prettylist list (ind ++ "␣␣␣")) ++
                                 ind ++ "\n"
       prettylist [] ind   = ""
       prettylist [v] ind = (pretty2 v (ind ++ "␣") (ind ++ "\\"))
       prettylist (v:rest) ind = (pretty2 v (ind ++ "|") (ind ++ "+")) ++
                                         (prettylist rest ind)
   in putStr (pretty2 t "" "")
```

prettyb draws a binary representation, NaryBin. prettyn draws a n-ary representation, NaryTree. Call as prettyb b and prettyn a