Recursive types

By using types with variables, recursive structures can be declared.

Declaration of a new tree

The following example defines a node of an binary ordered tree.

```
# type 'elemnttype nod = Empty
    | Nod of (' elementtype nod*' elementtype *' elementtype nod);;
type 'a nod = Empty | Nod of ('a nod * 'a * 'a nod)
```

It can be observed that the node contains an element of any type.

```
# let a= Nod (Empty,1,Empty);;
val a : int nod = Nod (Empty, 1, Empty)
```

Problem1.

Write the functions which insert an element in a tree, search an element in a tree and returns the elements of the tree in order and preorder.

Problem 2.

Given the expression trough the structure:

```
type expression = Value of float
   | Addition of (expression*expression)
   | Subtraction of (expression* expression)
   | Multiplication of (expression*expression)
   | Division of (expression* expression)
   | Fc1 of ((float->float)*expression)
   | Fc2 of ((float->float->float)*expression*expression);;
```

write the expression corresponding to: `6/2-2^3+cos(2*asin 1)` and a function with the signature (expression -> float) which evaluates any expression specified using the above structure

Problems