Tik-79.144 **Syksy 1999**
Logic in Computer Science: Foundations
Exercise 12
Solutions

1. a) The `mirror` predicate can be defined as follows:

```
mirror(leaf(V), leaf(V)).

mirror(tree(V, Left, Right), tree(V, NewLeft, NewRight) :-
   mirror(Left, NewRight),
   mirror(Right, NewLeft).
```

The first rule tells that a mirror image of a leaf node is the node itself. The second rule tells that we get a mirror image of a tree by recursively constructing mirror images of the two subtrees and switching them.

b) `tree(a, tree(b, leaf(c), leaf(d)), leaf(e))`

A `Prolog` interpreter processes the program and query as follows:

```
Welcome to SWI-Prolog (Version 3.1.0)
Copyright (c) 1993-1998 University of Amsterdam.  All rights reserved.

For help, use ?- help(Topic). or ?- apropos(Word).

%%% load the program  'mirror.pl'
1 ?- [mirror].
mirror compiled, 0.00 sec, 1,376 bytes.

Yes
%%% the query, 'trace' predicate prints the intermediate steps.
2 ?- trace, mirror(tree(a, tree(b, leaf(c), leaf(d)), leaf(e)), X).
T Call:  (  7) mirror(tree(a, tree(b, leaf(c), leaf(d)), leaf(e)), _G463)
T Call:  (  8) mirror(tree(b, leaf(c), leaf(d)), _G587)
T Call:  (  9) mirror(leaf(c), _G591)
T Exit:  (  9) mirror(leaf(c), leaf(c))
T Call:  (  9) mirror(leaf(d), _G590)
T Exit:  (  9) mirror(leaf(d), leaf(d))
T Exit:  (  8) mirror(tree(b, leaf(c), leaf(d)), tree(b, leaf(d), leaf(c)))
T Call:  (  8) mirror(leaf(e), _G586)
T Exit:  (  8) mirror(leaf(e), leaf(e))
T Exit:  (  7) mirror(tree(a, tree(b, leaf(c), leaf(d)), leaf(e)),
                       tree(a, leaf(e), tree(b, leaf(d), leaf(c))))

X = tree(a, leaf(e), tree(b, leaf(d), leaf(c)))

Yes
3 ?-
```

The answer tells us that the tree has a mirror image ("`Yes`") and the corresponding variable binding

```
X = tree(a, leaf(e), tree(b, leaf(d), leaf(c)))
```

c) In `Prolog` the order of the rules is significant. The variable symbol "_" is used to denote that we allow everything at that place.

```
member_tree(leaf(V), V).

member_tree(tree(V, _, _), V).

member_tree(tree(_, Left, _), V) :-
   member_tree(Left, V).

member_tree(tree(_, _, Right), V) :-
   member_tree(Right, V).
```

We can print out all members using a query of the form:

```
member_tree(tree(a, tree(b, leaf(c), leaf(d)), leaf(e)),
                           X)
```

An example run:

```
Welcome to SWI-Prolog (Version 3.1.0)
Copyright (c) 1993-1998 University of Amsterdam.  All rights reserved.

For help, use ?- help(Topic). or ?- apropos(Word).

1 ?- [tree].
tree compiled, 0.00 sec, 1,812 bytes.

Yes
2 ?- member_tree(tree(a, tree(b, leaf(c), leaf(d)), leaf(e)), X).

X = a ;

X = b ;

X = c ;

X = d ;

X = e ;

No
3 ?-
```