Overview

Goal: to explore tree manipulation in Standard ML.
Setting: Consider the polymorphic datatype

```
datatype 'a BinTree =
  Leaf of 'a
| Node of 'a BinTree * 'a BinTree
```

Example: An integer tree and its representation:

```
     /  \
    /    \
   /\     /\
  /  8  5  \
 /\        /\
4  7      3  11
```

```
val Tree1 = Node(
   Node (Node(Leaf(4),Leaf(7)),
         Leaf(8)),
   Node (Leaf(5),
         Node(Leaf(3),Leaf(11))))
```

Functions You Must Write (submit sml file)

print_tree that given a function that converts tree values into strings, and a binary tree, returns the values in that tree, with and between them.

```
- print_tree;
val it = fn : ('a -> string) -> 'a BinTree -> string
- print_tree Int.toString Tree1;
val it = "4␣and␣7␣and␣8␣and␣5␣and␣3␣and␣11" : string
```

deepest that returns the height (the maximal distance from the root to a leaf) of a binary tree, together with a list of the "deepest" nodes.

```
val it = fn : 'a BinTree -> int * 'a list
- deepest Tree1;
val it = (3,[4,7,3,11]) : int * int list
```

foldt that folds a binary tree, given a function combining two results, and a function converting a leaf into a result.

```
.. ('a * 'a -> 'a) -> ('b -> 'a) -> 'b BinTree -> 'a
- foldt op+ (fn x => x+1) Tree1;
val it = 44 : int
```

print_tree' with type and functionality as print_tree but defined using foldt and not by pattern matching.