Question 1 [13 marks]

This is "Tree 1" in the rubric.

(a) `t` is the type of a leaf in the tree.

(b) A tree of type `DecTree t` is in the `Show` class if `t` is in the `Show` class. To show a tree of type `t`, call `showb` with the tree and the empty list.

(c)
```
decode1 :: DecTree u -> [Int] -> (u, [Int])
decode1 (Val v) c = (v, c)
decode1 (Spl lt _) (0:c) = decode1 lt c
decode1 (Spl _ rt) (1:c) = decode1 rt c
```

Question 2 [12 marks]

This is "Tree 2" in the rubric.

(a) It can give an error if it is given a bit string that does not fully decode. This arises because `decode1` does not include a case for the empty list.

(b) It has to read the whole bit string to decide if the result is `Nothing`, instead of being able to use the first decoded symbol. (This was, by design, the most difficult question of the exam!)

(c) It was claimed to work for all types at the leaves, but with this declaration, it only works for strings at the leaves. The type of `decode1` and `decode` need to have `String` instead of the type variable `u`.

```
decode1 :: DecTree String ->  [Int] -> (String, [Int])
decode :: DecTree String -> [Int] ->  [String]
```

(d) Add the following to `decode1`

```
decode1 (Spl lt _)  [] = decode1 lt []
```

Question 3 [11 marks]

This is "MapFold Tree" in the rubric.

(a)
```
maptree :: (t1 -> t2) -> DecTree t1 -> DecTree t2
maptree f (Val t) = Val (f t)
maptree f (Spl lt rt) = Spl (maptree f lt) (maptree f rt)
```

(b) This question was designed to separate the A students from the rest.

```
treefold :: (t1 -> t2 -> t2) -> t2 -> DecTree t1 -> t2
treefold f b (Val t)  = f t b
treefold f b (Spl lt rt) = treefold f (treefold f b rt) lt
```

Question 4 [9 marks]

This is "Type Question" in the rubric.

(a) Here is a reduction (I added extra parentheses that were not required)

```
(((+).length) "abc") 4
   = ((+) (length "abc")) 4
   = ((+) 3) 4
   = 7
```

(b) There are lots of possible answers including:

- Each function can be tested and debugged separately.
- Each function gives the same answer each time it is called.
- Calling one function can never mess up another function call.

(c) There are lots of possible answers including:

- It helps you and other programmers read what you have done (good documentation).
- Defining the types of functions before they are written helps with designing the function.
- It helps the compiler find the source of an error when there is problem, because each function can be type checked independently.