CS 496: Homework Assignment 2
Due: 11 June 2023, 11:55pm

1 Assignment Policies

Collaboration Policy. This homework will be done individually: each student must hand in their own answers. It is acceptable for students to collaborate in understanding the material but not in solving the problems or programming. Use of the Internet is allowed, but should not include searching for existing solutions.

Under absolutely no circumstances code can be exchanged between students. Excerpts of code presented in class can be used.

Assignments from previous offerings of the course must not be re-used. Violations will be penalized appropriately.

2 Assignment

This assignment has two sections. The first one addresses the definition of a type `dTree` encoding *binary decision trees* in OCaml using variant types and then defining some simple operations on them. Two examples of such trees are given in Fig. 1. The second section presents a small application using `dTree`s which you are requested to implement.

3 The Type `dTree` in OCaml

1. Define `dTree`, a variant type in OCaml which encodes *binary decision trees* as depicted in Fig 1. It should have two constructors whose names should be `Leaf` and `Node`. Leaves should hold values of type `int` whereas nodes should hold values of type `char`. Finally, note that values of type `dtree` cannot be empty, they are either leaves or internal nodes.

```
type dTree = (* complete *)
```

2. Define two expressions, `tLeft` and `tRight`, of type `dTree` that represent each of the two trees in Fig. 1:

```
let tLeft = (* complete *)

let tRight = (* complete *)
```

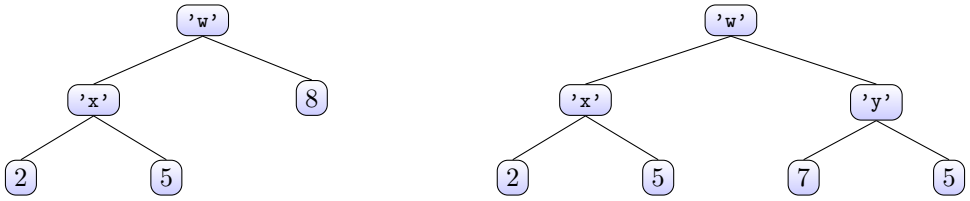


Figure 1: Sample values of type `dTree`

3. Implement the following functions indicating, for each one, its type. This is done by decorating your function definitions with type annotations, as seen in class. In the examples below, we use `tLeft` and `tRight` from the previous item.

   (a) `height`: that given a `dTree` returns its height. The height of a leaf should be 1. Eg.

   ```
   # height tLeft;;
   - : int = 3
   ```

   (b) `size`: that given a `dTree` returns its size. The size of a `dTree` consists of the number of inner nodes and leaves.

   ```
   # size tLeft;;
   - : int = 5
   ```

   (c) `paths`: that given a `dTree` returns a list with all the paths to its leaves. A path is a list of digits in the set $\{0, 1\}$ such that if we follow it on the tree, it leads to a leaf. The order in which the paths are listed is irrelevant. Eg.

   ```
   # paths tLeft;;
   - : int list list = [[0; 0]; [0; 1]; [1]]
   ```

   Also, when applied to a leaf, it should return `[[]]`.

   (d) `is_perfect`: that determines whether a `dTree` is *perfect*. A `dTree` is said to be *perfect* if all leaves have the same depth. Eg.

   ```
   # is_perfect tLeft;;
   - : bool = false
   # is_perfect tRight;;
   - : bool = true
   ```

   (e) `map`: that given the following arguments

   ```
   f: char -> char
   g: int -> int
   t: dTree
   ```

   returns a new `dTree` resulting from `t` by applying `f` to the characters in each node and `g` to the numbers in each leaf. For example,

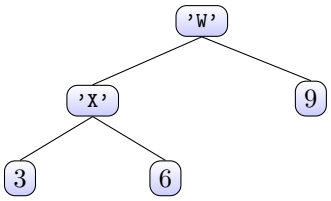
   ```
   map Char.uppercase_ascii (fun x -> x+1) tLeft
   ```

   should return an expression of type `dTree` representing the tree:

x	y	z	f
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

```
(['x';'y';'z'], [([0;0;0], 0);
                 ([0;0;1], 1);
                 ([0;1;0], 1);
                 ([0;1;1], 0);
                 ([1;0;0], 1);
                 ([1;0;1], 0);
                 ([1;1;0], 0);
                 ([1;1;1], 1)])
```

Figure 2: Binary function and its pair encoding in OCaml



Note that the internal nodes are now uppercase characters and the numbers in the leaves have been incremented by one.

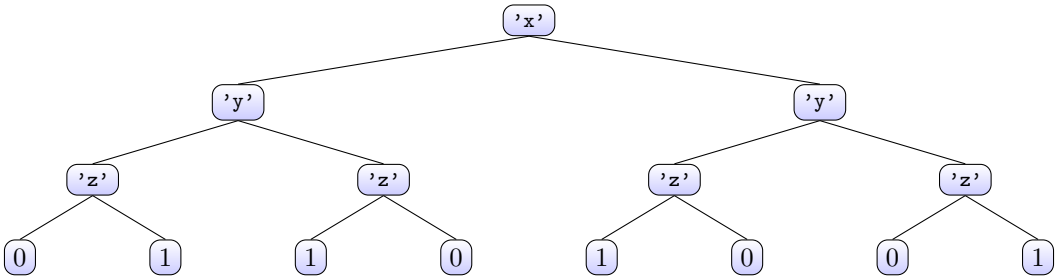
4 An Application of dTrees: Encoding Boolean Functions

A *boolean function* is a function from booleans to booleans. An example is given on the left in Fig. 2. This function f has three formal parameters, x, y and z. Each row indicates the value of f for the corresponding values of its parameters. For example, the first row states that `f 0 0 0` should return the value `0`. The need for the names of the formal parameters will become apparent below.

4.1 Encoding Boolean Functions Functions in OCaml

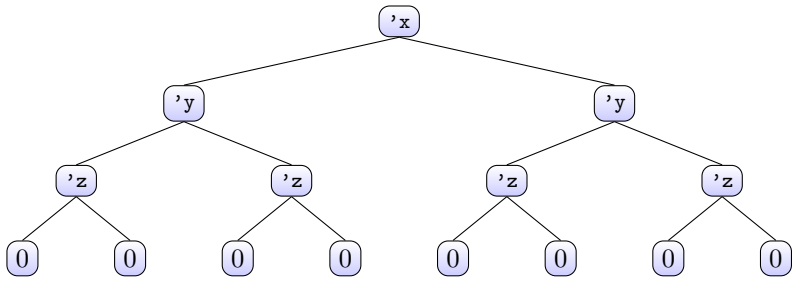
A boolean function may be encoded in OCaml in various ways. One naive way is just to use a pair consisting of two lists. The first list contains the names of the formal parameters. The second list contains the encoding of the outputs of the function for each possible value for the parameters. An example of this naive representation, which we might call the *pair-encoding*, is presented in Fig. 2 on the right.

However, there is an alternative encoding of boolean functions that uses the `dTree` type. We call this the *tree-encoding* of a boolean function. This encoding consists of a `dTree` in which each value of the function is given by a path in the tree (the leaf at the end of the path). For example, the tree-encoding of f is:



The aim of this exercise is to define a function `bf_to_dTree` that takes a pair-encoding of a boolean function and returns its tree-encoding.

4. Define `list_to_tree`, a function that given a list of characters `l`, creates a tree in which the symbols of an inner node at level n corresponds to the n-th element in `l`. The value of all of its leaves may be set to 0. E.g. `list_to_tree ['x';'y';'z']` produces the `dTree` representing:



5. Define `replace_leaf_at`, a function that given a tree `t` and a graph for a function `f`, replaces all the leaves in `t` by the value indicated in the graph of the function. The graph of a pair-encoded function is its second component. For example, the graph of the pair-encoded function in Fig. 2(right) is its right component, namely:

```
[([0;0;0], 0);
 ([0;0;1], 1);
 ([0;1;0], 1);
 ([0;1;1], 0);
 ([1;0;0], 1);
 ([1;0;1], 0);
 ([1;1;0], 0);
 ([1;1;1], 1)]
```

6. Define a function `bf_to_dTree` that takes a pair-encoding of a boolean function and returns its tree-encoding. Note that, depending on the order in which the formal parameters are processed, there may be more than one possible tree-encoding for a given pair-encoding. You may return any of them.

5 Submission instructions

Submit a file `hw2.ml` through Canvas. Your grade will be determined as follows:

Exercise	Grade
1	1
2	1
3(a)-(e)	5 (1 each)
4	1
5	1
6	1