COMP 3958: Lab 3

Submit a zip file named `lab3.zip` containing your 2 source files: `part1.ml` and `kvtree.ml`. Unless otherwise indicated, you may use functions in `Stdlib`, but not in its submodules. (However, you may use `Fun.flip` in one instance.) As before, your files must compile. Otherwise, you may fail to get credit. Maximum score: 14

1. (a) Implement from basics a function `digits` that returns the list of digits in an integer. For example, `digits 3276` returns the list `[3; 2; 7; 6]`.

   (b) Using either `List.fold_left` or `List.fold_right`, implement a function `int_of_digits` that returns an integer from a list of its digits. For example, `int_of_digits [3; 2; 7; 6; 8]` returns 32768. Note that leading zeros in the argument list do not change the returned integer.

   Put your code in a file named `part1.ml`.

2. A binary search tree is usually used to store key-value pairs and we typically search for a particular key to find the corresponding value.

   Modify the binary search tree from class to use 2 type parameters – one for the key and the other for the value. We'll call the new tree `kvtree` (for key-value tree). Its type is `('k, 'v) kvtree`.

   Implement the following functions for `kvtree` (they are analogous to the ones for `bstree`):

```
val kvtree_empty : ('k, 'v) kvtree
val kvtree_is_empty : ('k, 'v) kvtree -> bool
val kvtree_insert : cmp:('k -> 'k -> int) -> 'k -> 'v
                    -> ('k, 'v) kvtree -> ('k, 'v) kvtree
val kvtree_find_opt : cmp:('k -> 'k -> int) -> 'k -> ('k, 'v) kvtree
                      -> 'v option
val kvtree_delete : cmp:(k' -> 'k -> int) -> 'k -> ('k, 'v) kvtree
                    -> ('k, 'v) kvtree
val kvtree_of_list : cmp:('k -> 'k -> int) -> ('k * 'v) list
                     -> ('k, 'v) kvtree
```

   - Each of the above functions has a labeled argument (`cmp`) that specifies a comparison function used to compare keys. Its purpose is similar to the `cmp` argument in `ListLabels.sort`.
   - For `kvtree_insert`, if the key is already in the tree, the corresponding value is updated to the new value;
   - The `kvtree_find_opt` function is the replacement for the `bstree_mem` function. It returns the corresponding value if there is one. (However, note that its return type is `'v option`.) Note also that OCaml may infer a more general type for `kvtree_find_opt` than that shown above. (You can use type annotations to restrict it to the type shown.)

   You may need to implement additonal helper functions. To facilitate testing, we need to standardize the internal representation of kvtrees. Here are 2 examples of kvtrees: `L` (an empty tree) and `N ("homer", 45, L, L)`. (Note that the `N` constructor takes 4 arguments.) Name your file `kvtree.ml`.