# CSE 305: Pattern Matching and Recursion

Due: Wednesday, October 9, at 11:59 pm

## 1 Overview

The solution to this assignment has to be in Ocaml.

As a warm up you are required to implement the following two functions:

1. . `write_list_int: string -> int list -> unit`.

   This function writes a list of integers, provided as second argument, in a file, whose name is given as string in the first argument. The numbers have to be written one per line following the order left to right in the list. The function then returns `unit`. You can reuse code of other assignments to solve this task.

2. `read_list_int_rev: string -> int list`.

   This function takes the name of a file (which contains one integer per line) and returns a list of all the integers in reversed order. That is, if the file contains
   1
   2
   3
   then the function should return [3;2;1]. Again, you are free to reuse code of other assignments to solve the task.

Now that you warmed up, you are asked to implement some basic functions with a specified behavior and type (or signature) working with a data type representing *biltree* which is defined as follow:

```
type biltree = B of bool   | I of int
             | L of int list | T of biltree * char * biltree
```

This definition is also provided in the file *biltree.ml* which you can use as a basis for your development.

A biltree element can be a bool, an int, a list of int or a triple containing a left biltree, a character, and a right biltree (notice that the definition of `biltree` is recursive). Here some examples of `biltree` values:

```
ex1 = T (T (B true, 'a' , T (I (-34), 'b', L [-21; 53; 12])), 'c', T (I (-18), 'd' ,
      B true))
ex2 = T (T (T (T (I 31, 'h', L [9; 34; -45]), 'e', L [70; 58; -36; 28]), 'l', I 3),
      'l', T (I 2, 'o', I 49))
ex3 = T (T (T (L [9; 4; -1; 0; -5], 'c', B true), 's', B true), '3', T (B false,
      '2', I (-3)))
```

We will use these examples in the sequel to illustrate the functions that you are required to implement.

## 2 Functions over biltrees

1. `count_nodes: (b: biltree) -> int`.

   The first function you need to implement is `count_nodes` whose behavior is to recursively traverses the input biltree `b` and counts how many nodes it contains. The `B b, I i, L l` can be considered as the leaf nodes and `T(tl, c, tr)` is an internal node with value `c`, left subtree `tl` and right subtree `tr`. `count_nodes` should count all these nodes
   For example, if we run `count_chars` on our examples we have:

   ```
   count_nodes ex1 = 9
   count_nodes ex2 = 11
   count_nodes ex3 = 9
   ```

2. `global_and: (b: biltree) -> bool option`.

   The second function you need to implement is `global_and` whose behavior is to compute the logical *and* (in OCaml &&) of all the booleans inside a biltree value `b`. There are two "corner cases" that your solution needs to consider: 1) if there is exactly one Boolean in the biltree `b` then `global_and` should return this value as result, 2) if there is no Boolean in `b` then, since we cannot compute an and, `global_and` should return a value `None` asserting that no information is available. For example, if we run `global_and` on our examples we have:

   ```
   global_and ex1 = Some true
   global_and ex2 = None
   global_and ex3 = Some false
   ```

3. `sum_lists: (b: biltree) -> biltree`.

   The third function you need to implement is `sum_lists` which should traverse a biltree `b` and replace each int list node by a int node containing a value corresponding to the sum of all the elements of the list.
   For example, if we run `sum_lists` on our examples with some auxiliary anonymous functions we have:

   ```
   sum_lists ex1 =
      T (T (B true, ’a’, T (I (-34), ’b’, I 44)), ’c’, T (I (-18), ’d’, B true))
   sum_lists ex2 =
      T (T (T (T (I 31, ’h’, I (-2)), ’e’, I 120), ’l’, I 3), ’l’,
      T (I 2, ’o’, I 49))
   sum_lists ex3 =
      T (T (T (I 7, ’c’, B true), ’s’, B true), ’3’, T (B false, ’2’, I (-3)))
   ```

   You may find helpful here to use the function `List.fold_left` or the function `reduce` we saw in class.

4. `f_on_all_ints: (f: int -> int) -> (p: biltree) -> biltree`.

   The fourth function you need to implement is `f_on_all_ints` which should traverse a biltree `b` and apply `f` on all the int nodes in `b`. For example, if we run `f_on_all_ints` on our examples we have:

   ```
    f_on_all_ints (fun t -> t - 10) ex1 =
         T (T (B true, 'a', T (I (-44), 'b', L [-31; 43; 2])), 'c',
         T (I (-28), 'd', B true))
    f_on_all_ints (fun t -> 2 * t) ex2 =
         T(T (T (T (I 62, 'h', L [18; 68; -90]), 'e', L [140; 116; -72; 56]),
         'l', I 6), 'l', T (I 4, 'o', I 98))
    f_on_all_ints (fun t -> t * t) ex3 =
         T (T (T (L [81; 16; 1; 0; 25], 'c', B true), 's', B true), '3',
         T (B false, '2', I 9))
   ```

   You may find helpful here to use the function `List.map` to implement your solution.

5. `tostring_mlr: (b: biltree) -> string`.

   The fifth function you need to implement is `tostring_mlr` which converts a biltree `b` into a string, following a mid first then left and then right subtree order traversal of the tree. That means that when you are converting to string a biltree element of the form `T(tl, c, tr)` you need to concatenate the recursive calls as `(Char.escaped c) ^ (tostring_mlr tl) ^(tostring_mlr tr)` . Here `Char.escaped` is a library function to convert a character in a string.

   For example, if we run `tostring_mlr` on our examples we have:

   ```
   tostring_mlr ex1="catrueb-34-215312d-18true"
   tostring_mlr ex2="lleh31934-457058-36283o249"
   tostring_mlr ex3="3sc94-10-5truetrue2false-3"
   ```

   You may find convenient in your solution to first define a function `string_of_intlist` converting a list of integers into a string which can then be used in the definition of `tostring_mlr`. The function `string_of_intlist` can be implemented using the functions `List.fold_left` or `reduce` and `List.map` that we saw in class.

## 3 Submission Instructions

Late submissions will not be accepted. You can use Gradescope to confirm your program adheres to the specification outlined. Only your last Gradescope submission will be graded for you final grade. This means that you can submit as many times as you want before the deadline. If you have any questions please ask well before the due date on Piazza.

### 3.1 GradeScope submission instructions

When you log into Gradescope and pick the CS 320 course you will see an assignment called biltree. You will need to submit a solution written in Ocaml to this part to have full credit for the assignment. The total of points for this assignment is: 50. Your program must be an .ml file.

### 3.2 Tips on GradeScope submission

Be sure to check:

- your function should have exactly the same signature as required.
- your file should named as biltree.ml
- when you use some library function, make sure you open this libiary at the begining of your program
- please do not include any `#use biltree.ml` or `;;` in your code.

### 3.3 Additional Resources

For information on how to run OCaml please see the following references:

- `https://caml.inria.fr/pub/docs/manual-ocaml/stdlib.html`
- `https://caml.inria.fr/pub/docs/manual-ocaml/`

You will find resources for these languages on the Piazza course web site, under the Resources page.