T-501-FMAL Programming languages, Practice class 8
Spring 2021

1. Infer the types for these expressions, first manually and then use the type inferrer from HigherFunInfer.fs or F#'s type inference to check your answers. Explain the results.

   (let is meant as recursive where appropriate. Take this into account when giving these expressions to F#.)

```
let f x = false in f

let f x = (let g y = y in g false) in f

let f x = (let g y = y in g (g 3 < 4)) in f

let f x = (let g y = if true then y else x in g false) in f

let f x = (let g y = if true then y else x < 3 in g false) in f

let f x = x + 3 in let g h = h (f 8) in g

let rec f x = f x + 1 in f

let rec f x = f (x + 1) in f
```

2. Write expressions in our higher-order language or in F# for which type inference returns these types:

```
bool -> bool

int -> bool

(int -> bool) -> int

'a -> 'b -> 'b

('a -> 'b) -> ('b -> 'c) -> 'a -> 'c

('a -> 'b) -> ('a -> 'b -> 'c) -> 'a -> 'c

('a -> 'a -> 'b) -> 'a -> 'b

('a -> 'e -> 'd) -> ('b -> 'c -> 'e) -> 'a -> 'b -> 'c -> 'd
```

3. Consider extending the language of HigherFun.fs and HigherFunInfer.fs with pair types as follows:

```
type expr =
    | ...
    | MkPair of expr * expr     // (e1, e2)
    | Fst of expr               // fst e
    | Snd of expr               // snd e

type value =
    | ...
    | P of value * value        // (v1, v2)

type typ =
    | ...
    | Pair of typ * typ         // t1 * t2
```

Now extend the evaluator and the type inferrer (in the code for the latter you need to update the functions `unify` and `infer` and a number of auxiliary functions).