HW 4: Solutions

1 Theoretical Questions (points)

Problem 1: (25 points)

Give a complete type derivation for the following typing judgment.

```
let rec f = fun x -> fun n -> if n <= 0 then [] else x::(f x (n - 1))
       in (f 3 2, f "a" 4) : int list * string list
```

As a suggestion for formatting, you may want to name subtrees of the proof and write them out separately. Note, we are asking for a type derivation not the intermediate states of a type inferencing algorithm.

Solution:

Let

Γ_1 = {f : 'a -> int -> 'a list}
Γ_2 = {f : ∀ 'a.'a -> int -> 'a list}
Γ_3 = {f : 'a -> int -> 'a list, x : 'a}
Γ_4 = {f : 'a -> int -> 'a list, x : 'a, n : int}

Let subtree *RecAppTree* be defined by

$$\dfrac{\dfrac{}{\Gamma_4 \vdash \texttt{f} : \texttt{'a -> int -> 'a list}}\textsc{Var} \quad \dfrac{}{\Gamma_4 \vdash \texttt{x} : \texttt{'a}}\textsc{Var}}{\Gamma_4 \vdash \texttt{f x} : \texttt{int -> 'a list}}\textsc{App} \quad \dfrac{\dfrac{}{\Gamma_4 \vdash \texttt{n} : \texttt{int}}\textsc{Var} \quad \dfrac{}{\Gamma_4 \vdash \texttt{1} : \texttt{int}}\textsc{Const}}{\Gamma_4 \vdash \texttt{(n-1)} : \texttt{int}}\textsc{Arith}}{\Gamma_4 \vdash \texttt{(f x (n-1))} : \texttt{'a list}}\textsc{App}$$

Let subtree *ConsTree* be defined by

$$\dfrac{\dfrac{}{\Gamma_4 \vdash \texttt{(::)} : \texttt{'a -> 'a list -> 'a list}}\textsc{Const}\ \texttt{'a -> 'a list -> 'a list instance of ∀ 'a.'a -> 'a list -> 'a list} \quad \dfrac{}{\Gamma_4 \vdash \texttt{x} : \texttt{'a}}\textsc{Var}}{\Gamma_4 \vdash \texttt{(::) x} : \texttt{'a list -> 'a list}}\textsc{App} \quad \dfrac{\textit{RecAppTree}}{\Gamma_4 \vdash \texttt{(f x (n-1))} : \texttt{'a list}}}{\Gamma_4 \vdash \texttt{x::(f x (n-1))} : \texttt{'a list}}\textsc{App}$$

Let subtree *FuncTree* be defined by

$$\dfrac{\dfrac{\dfrac{}{\Gamma_4 \vdash \texttt{n : int}}\textsc{Var}\quad\dfrac{}{\Gamma_4 \vdash \texttt{0 : int}}\textsc{Const}}{\Gamma_4 \vdash \texttt{n <= 0 : bool}}\textsc{Rel}\quad\dfrac{\texttt{'a list instance of } \forall\ \texttt{'a.'a list}}{\Gamma_4 \vdash \texttt{[] : 'a list}}\textsc{Const}\quad\dfrac{\textit{ConsTree}}{\Gamma_4 \vdash \texttt{x::(f x (n-1)) : 'a list}}}{\Gamma_4 \vdash \texttt{if n <= 0 then [] else x::(f x (n-1)) : 'a list}}\textsc{If}$$

$$\dfrac{\dfrac{\Gamma_4 \vdash \texttt{if n <= 0 then [] else x::(f x (n-1)) : 'a list}}{\Gamma_3 \vdash \texttt{fun n -> if n <= 0 then [] else x::(f x (n-1)) : int -> 'a list}}\textsc{Fun}}{\Gamma_1 \vdash \texttt{fun x -> fun n -> if n <= 0 then [] else x::(f x (n-1)) : 'a -> int -> 'a list}}\textsc{Fun}$$

Let subtree *IntAppTree* be defined by

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma_2 \vdash \texttt{f : int -> int -> int list}}\textsc{Var}\quad\dfrac{\texttt{int -> int -> int list instance of } \forall\ \texttt{'a. 'a -> int -> 'a list}}{\Gamma_2 \vdash \texttt{3 : int}}\textsc{Const}}{\Gamma_2 \vdash \texttt{f 3 : int -> int list}}\textsc{App}\quad\dfrac{}{\Gamma_2 \vdash \texttt{2 : int}}\textsc{Const}}{\Gamma_2 \vdash \texttt{f 3 2 : int list}}\textsc{App}$$

Let subtree *StringAppTree* be defined by

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma_2 \vdash \texttt{f : string -> int -> string list}}\textsc{Var}\quad\dfrac{\texttt{string -> int -> string list instance of } \forall\ \texttt{'a. 'a -> int -> 'a list}}{\Gamma_2 \vdash \texttt{``a'' : string}}\textsc{Const}}{\Gamma_2 \vdash \texttt{f ``a'' : int -> string list}}\textsc{App}\quad\dfrac{}{\Gamma_2 \vdash \texttt{4 : int}}\textsc{Const}}{\Gamma_2 \vdash \texttt{f ``a'' 4 : string list}}\textsc{App}$$

Let *AppTree* be defined by

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma_2 \vdash \texttt{(,) : int list -> string list -> int list * string list}}\textsc{Const}\quad\dfrac{\texttt{int list -> string list -> int list * string list instance of } \forall\ \texttt{'a 'b. 'a -> 'b -> ('a, 'b)}}{\Gamma_2 \vdash \texttt{f 3 2 : int list}}\textsc{App}}{\Gamma_2 \vdash \texttt{(,) (f 3 2) : string list -> int list * string list}}\quad\dfrac{\textit{StringAppTree}}{\Gamma_2 \vdash \texttt{f ``a'' 4 : string list}}\textsc{App}}{\Gamma_2 \vdash \texttt{(f 3 2, f ``a'' 4) : int list * string list}}\textsc{App}$$

Finally, the type derivation for the given expression is

$$\dfrac{\dfrac{\textit{FuncTree}}{\Gamma_1 \vdash \texttt{fun x -> fun n -> if n <= 0 then [] else x::(f x (n-1)) : 'a -> int -> 'a list}}\quad\dfrac{\textit{AppTree}}{\Gamma_2 \vdash \texttt{(f 3 2, f ``a'' 4) : int list * string list}}}{\vdash \texttt{let rec f = fun x -> fun n -> if n <= 0 then [] else x::(f x (n-1)) in (f 3 2, f "a" 4) : int list * string list}}\textsc{Rec}$$

2 Machine Problems

Please see `solution.ml`.