Exercises:

(ϕ denotes the empty typing environment.)

(1) $\phi \vdash$ if $1 \&\& (-1)$ then 0 else 1 : Int

(2) $\phi \vdash -1$ @ Nil[Int] : List[Int]

(3) $\phi \vdash$ let x:Int $= 0$ in let x:List[Int] = Nil[Int] in x : List[Nil]

(4) $\phi \vdash$ (3@Nil[Int]) @ Nil[List[Int]] : ?

Fill in ? so that the expression is well-typed.

(5) $\phi \vdash (\lambda x{:}\text{Int}.\ \lambda x{:}\text{List[Int]}.\ x)$: Int → List[Int] → ?

Fill in ? so that the expression is well-typed.

(6) $\phi \vdash$ fix $(\lambda x{:}\text{Int}.\ x)$: Int

(7) $\phi \vdash$ fix $(\lambda f{:}\text{Int}\to\text{Int}.\ \lambda x{:}\text{Int}.\ x)$: ?

(i) Fill in ? so that the expression is well-typed.

(ii) $(\lambda f{:}\text{Int}\to\text{Int}.\ \lambda x{:}\text{Int}.\ x)$ is the generator of some function g. What is g? what does g do?

* (8) $\phi \vdash$ fix $(\lambda f{:}\text{Int}\to\text{Int}.\ \lambda x{:}\text{Int}.$ if $x=0$ then 1 else $f(n-1) * x)$: ?

(i) This is also the generator of some function g.

What is g?

(ii) Fill in ? so that the expression is well-typed.

(9) Show that under ϕ, $(\lambda f{:}\text{Int}\to\text{Int}.42)$(if 1 then 2 else 3)

is ill-typed. Which rule gets stuck?

* (10) Show that if we introduce a new type $\bot$

with no constructors and no evaluation rules at all,

there still exists a term in λ^+ that has type $\bot$.

(Hint: See exercise 6).

(11) let x:Int = 1 in x + (let x:List[Int] = x@Nil[Int] in ! x)

λ^+ extensions :

1) Concrete syntax :

- $\lambda x : T . e$
- let $x : T = e_1$ in e_2
- Nil [T]

T can be Int , List[T] , T → T

E.g.

2) Abstract syntax :

- Lambda of string * typ option * expr
- LetBind of string * typ option * expr * expr
- ListNil of typ option

where typ represents T .

★ In HW5, you may assume all type annotations are provided.
That is, a typ option expression must be Some(t).