--HW4 for FoPL

--Stu:金泽文 No:PB15111604

1.

quicksort :: (Ord a) => [a] -> [a]

--I don't think lesser and greater are functions

lesser :: [Ord a]

greater :: [Ord a]

--quicksort [] = []

--=> it must be a list

--quicksort (p:xs) = (bla) ++ [p] (bla)

--=> it must be a list of 'p's

--filter (<p) xs

--=> 'p' must can be compared and ordered and should be of the type which is a member of Ord

2.

(a)

(Eq t, Num t) => (t -> t) -> t -> t

--the use of 'case of x' => x can be tested with '=' symbol, so t i a member of Eq

--the use of 'x * ' and 'x - 1' => t is a member of Num

--the use of 'x * g (x-1)' =>, g takes a 't' type argument and returns a 't' type

```
result, so g :: t -> t

    --finally, it should be

        (Eq t, Num t) => (t -> t) -> t -> t
```

(b)

```
    (a -> a) -> a

    --assume f :: a -> b , y :: (a->b) -> c

    -- f (y f)    => (y f) :: a

    --            => c = a

    --            => y f :: a

    --            => f (y f) :: a

    --            => b = a

    --            => y :: (a -> a) -> a
```

(c)

```
    fibRec g n = case n of

                        0 -> 0

                        1 -> 1

                        n -> (g (n - 1)) + (g (n - 2))

    --y fibRec n = fibRec (y fibRec) n

    --             = case n of

                        0 -> 0

                        1 -> 1

                        n -> ((y fibRec) (n - 1)) + ((y fibRec) (n - 2))
```

(d)

i.

```
--   y(f) = (\g -> f(g g) ) (\g -> f(g g))
--           = f((\g -> f(g g)) (\g -> f(g g)))
--   f(y(f)) = f((\g -> f(g g) ) (\g -> f(g g)))
--   y(f) = f(y(f))
```

ii.

```
--because the compiler will check the type with the type inference, while
--   y f = (\g -> f (g g) ) (\g -> f (g g))
     --   = f ((\g -> f (g g)) (\g -> f (g g)))
     --   = f (f ((\g -> f (g g)) (\g -> f (g g))))
     --   = f (f (f ((\g -> f (g g)) (\g -> f (g g)))))
     --   = ...
-- the type inference won't stop, so error occurred
```

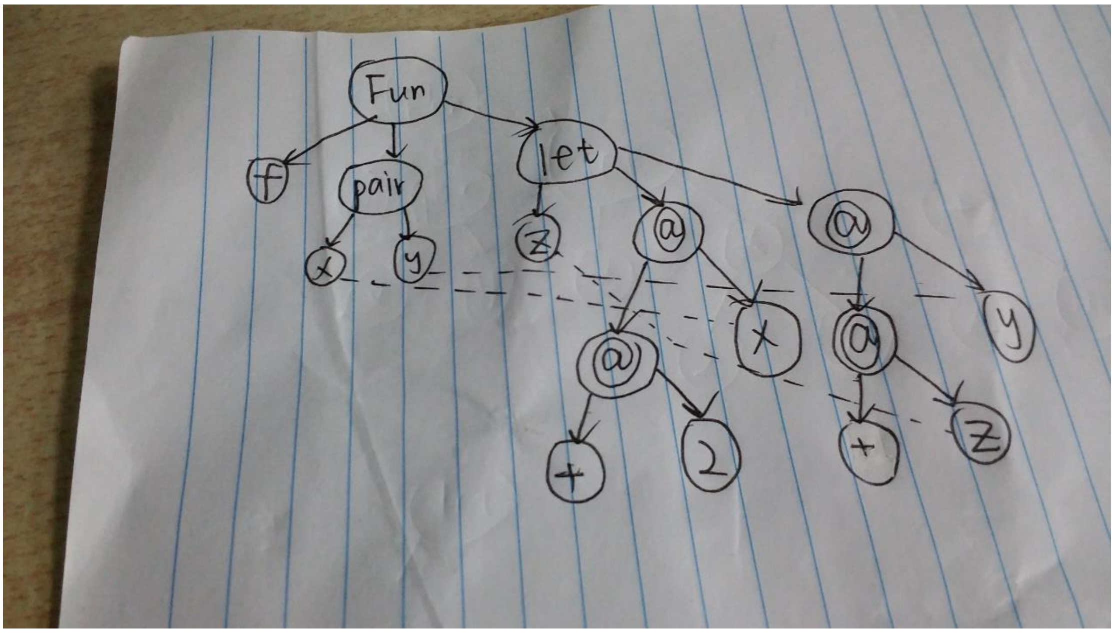
(e)

```
reduceRec g f l = case l of
                    [] -> undefined
                    [x] -> x
                    (x:xs) - > f x (g f xs)
--
     y reduceRec f l = reduceRec (y reduceRec) f l
```

```
                     = case l of
                          [] -> undefined
                          [x] -> x
                          (x:xs) - > f x ((y reduceRec) f xs)
reduce f l = case l of
                     [] -> undefined
                     [x] -> x
                     (x:xs) - > f x (reduce f xs)
```

3.

（c）如图。



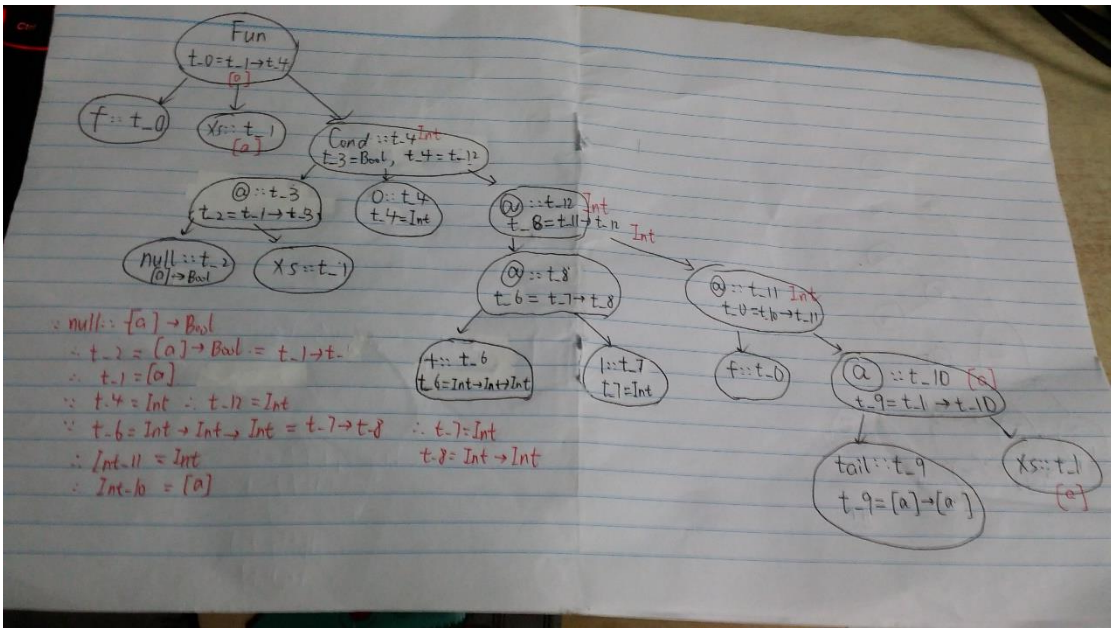
（d）

i.没太理解题的意思，假设题意要求的是：说明 Cond 节点生成的约束。

答：Cond 节点左下方的“@”节点类型应为 Bool，

Cond 节点下方和右下方节点类型应该相同。

ii.注释如图



∵ null:: [a]→Bool

∴ t_2 = [a]→Bool = t_1→t_

∴ t_1 = [a]

∵ t_4 = Int ∴ t_12 = Int

∵ t_6 = Int→Int→Int = t_7→t_8 ∴ t_7=Int

t_8= Int→Int

∴ Int_11 = Int

∴ Int_10 = [a]

iii. f::[a]->Int

因为 null::[a]->Bool，所以 t_1（即 xs 类型）为[a],

因为 0::Int,所以 Cond 的类型应该为其 then 和 else 的类型，也就是 Int

所以 f:: [a]->Int。

4.

（a）

因为 concatS :: (String, String)-> String, t_9 是 pair，

所以 t_10 是 String，t_7，t_1 是 String。

因为 showI :: Int->String.

所以 t_2 是 Int

所以，t_3=（String，Int）

所以 g :: (String, Int) -> String

（b）

foldright : : ((a, b) -> b) –>b -> [a] -> b

（c）

因为 g::(Int -> String) -> String

（d）

g __（n，s）_____ = concatS _____（showI n，s）____________

5.

(a)

dCompInt = \x y ->(if (x > y) then GT else (if (x == y) then EQ else LT))

compList (x:xs) (y:ys) =

if (((?=)__x_y________________________) /= EQ)

then ((?=)__x_y_______________________________)

else ((?=)__xs___ys_____________________________)

(b)

(?=) (\(x1,x2) (y1,y2) -> if ((x1 ?= y1) /= EQ) then (x1 ?= y1) else (x2 ?= y2)) _________________ (length "Hello","Hello") (length "World","World")

(?=) ___dCompareInt__________ _____length “ Hello ” ___________ __length “World” _____________

(?=) (\(x:xs) (y:ys)_if ((x ?= y) /= EQ) then (x ?= y) else (xs ?= ys))

_ "Hello" "World"

(?=) dCompChar ’ H’ ’ W’

(c)

Because the type of the “length” is the (Foldable t)=> t -> Int, while the tuple “()” is not a member of the Foldable. So, from the definition of the “?=” we can infer that x and y can only be the lists of Int or Char, so my answer is

f :: [a] -> [a] -> Ordering 这里的 a 是 Int 或 Char(还没找到怎么表示 Int 和 Char 并集的类型类)

6.

(a)

MyEqD 指的是 dictionary，=== function 表示对应的函数

(b)

分别填入 MyEq a 、Tree a 、v1===v2&&tl1===tl2&&tr1===tr2

(c)

分别填入 MyEqD a->MyEqD Tree a 、(===)d v1 v2 、 ((===)d v1 v2)、&&(myEqtree tl1 tl2)、 &&(myEqtree tl1 tl2)

(e)

分别填入(MyEqD a)->a->a->String、 d t1 t2 、 ((===)d t1 t2) 、 dMyEqInt