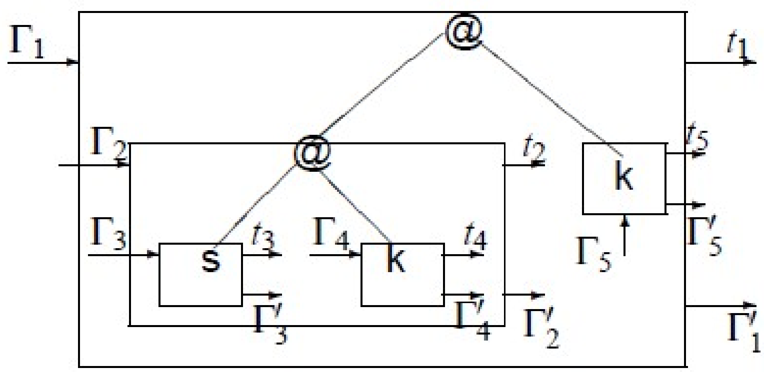
POPL Assignment 3 (Large)

---

1. Consider inferencing the type of the term shown below using the Hindley-Milner algorithm.

```
let
    k = \x y -> x
in s k k
```

Show the type inferencing of the body of the let expression by specifying the type environments Γ_i and Γ'_i, substitutions θ_i, and the types t_i in the figure below. Assume s is a builtin function with type $s :: \forall \alpha\beta\gamma.(\alpha \to \beta \to \gamma) \to (\alpha \to \beta) \to \alpha \to \gamma$ (15 Marks)



2. A recursive definition for **concat** is:

concat [ ] $\quad$ = [ ]
concat (x:xs) = **x ++ concat xs**

Haskell uses Hindley-Milner style type inferencing to infer the type of **concat**. Assume the following is present in the initial type environment:

$$(++) :: \forall \alpha \cdot [\alpha] \to [\alpha] \to [\alpha]$$

Explain through box-diagrams the process of finding type of concat. Your solution must indicate the input type assumptions (Γ), unification required (θ) and the discovered type (t). All other steps that are important to infer the type of **concat** must be added. (What is the type of empty list [ ] ?) (25 Marks)

3. Suppose we had a type system similar to the one studied in the class except that M-Abs and M-App are substituted by the rules P-App and P-Abs. Show that the term $M = \lambda x.xx$ is typable in this type system by finding a type σ for it and then proving $M :: \sigma$. (15 Marks)

   Use the definitions of P-App and P-Abs shown below:

$$\frac{\Gamma \vdash M :: \sigma \to \tau \qquad \Gamma \vdash N :: \sigma}{\Gamma \vdash M\ N :: \tau} \qquad \text{(P-App)}$$

$$\frac{\Gamma, x :: \sigma \vdash M :: \tau}{\Gamma \vdash \lambda x.M :: \sigma \to \tau} \qquad \text{(P-Abs)}$$

4. For every C_1 , C_2 and α, state whether (i) α is a unifier of C_1 and C_2, and (ii) α is the MGU of C_1 and C_2. If α is not the MGU, give one. (15 Marks)

#	C1	C2	α
(a)	$p(a, f(y), z)$	$q(x, f(f(v)), b)$	$\{x := a, y := f(b), z := b\}$
(b)	$q(x, h(a, z), f(x))$	$q(g(g(v)), y, f(w))$	$\{x := g(g(v)), y := h(a, z), w := x\}$
(c)	$q(x, h(a, z), f(x))$	$q(g(g(v)), y, f(w))$	$\{x := g(g(v)), y := h(a, z), w := g(g(v))\}$
(d)	$r(f(x), g(y))$	$r(z, g(v))$	$\{x := a, z := f(a), y := v\}$