Question 1 [50 points]: "If it's 'dynamic', it *must* be better."

The following rules define an environment-based semantics for lexically-scoped functions, Lam, and dynamically-scoped functions, Ds-lam.

$\boxed{env \vdash e \Downarrow v}$ Under environment env, expression e evaluates to value v

$$\frac{}{env \vdash (\text{Num } n) \Downarrow (\text{Num } n)}\ \text{Env-num} \qquad \frac{env \vdash e1 \Downarrow (\text{Num } n1) \qquad env \vdash e2 \Downarrow (\text{Num } n2)}{env \vdash (\text{Add } e1\ e2) \Downarrow (\text{Num } n1{+}n2)}\ \text{Env-add}$$

$$\frac{env \vdash e1 \Downarrow v1 \qquad x{=}v1, env \vdash e2 \Downarrow v2}{env \vdash (\text{Let } x\ e1\ e2) \Downarrow v2}\ \text{Env-let}$$

$$\frac{lookup(env, x) = e}{env \vdash (\text{Id } x) \Downarrow e}\ \text{Env-id} \qquad \frac{lookup(env, x)\ \text{undefined}}{env \vdash (\text{Id } x)\ \text{unknown-id-error}}\ \text{Env-unknown-id}$$

$$\frac{}{env \vdash (\text{Lam } x\ e1) \Downarrow (\text{Clo } env\ (\text{Lam } x\ e1))}\ \text{Env-lam} \qquad \frac{env_{old} \vdash e \Downarrow v}{env \vdash (\text{Clo } env_{old}\ e) \Downarrow v}\ \text{Env-clo}$$

$$\frac{env \vdash e1 \Downarrow (\text{Clo } env_{old}\ (\text{Lam } x\ eB)) \qquad env \vdash e2 \Downarrow v2 \qquad x{=}v2, env_{old} \vdash eB \Downarrow v}{env \vdash (\text{App } e1\ e2) \Downarrow v}\ \text{Env-app}$$

$$\frac{}{env \vdash (\text{Ds-lam } x\ e1) \Downarrow (\text{Ds-lam } x\ e1)}\ \text{Env-ds-lam} \qquad \frac{env \vdash e1 \Downarrow (\text{Ds-lam } x\ eB) \qquad env \vdash e2 \Downarrow v2 \qquad x{=}v2, env \vdash eB \Downarrow v}{env \vdash (\text{App } e1\ e2) \Downarrow v}\ \text{Env-ds-app}$$

Assume that $lookup(env, x)$ returns the **leftmost** binding of x. For example:

$$lookup\Big((x{=}(\text{Num } 2), x{=}(\text{Num } 1), \emptyset), x\Big) = (\text{Num } 2)$$

Consider the following expression, shown in concrete syntax (left) and in abstract syntax (right).

```
{Let y 100
     {Let f {Let y 10
                 {Lam x {+ x y}}}
            {Let y 2
                 {App f y}}}}
```

```
(Let y (Num 100)
  (Let f (Let y (Num 10)
               (Lam x (Add (Id x) (Id y))))
       (Let y (Num 2)
            (App (Id f) (Id y)))))
```

Q1a [10 points] Complete the **abstract syntax tree** for the above expression. ⟹