## Question 1 [55 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.

Under environment env, expression e evaluates to value 
$$v$$

$$\frac{env \vdash e \downarrow v}{env \vdash (num \ n) \Downarrow (num \ n)} \xrightarrow{Env-num} \frac{env \vdash e1 \Downarrow (num \ n1)}{env \vdash (add \ e1 \ e2) \Downarrow (num \ n1 + n2)} \xrightarrow{Env-add} \frac{env \vdash e1 \Downarrow v1}{env \vdash (with \ x \ e1 \ e2) \Downarrow v2} \xrightarrow{Env-with} \frac{env \vdash e1 \Downarrow v1}{env \vdash (id \ x) \Downarrow e} \xrightarrow{Env-id} \frac{lookup(env, x) \ undefined}{env \vdash (id \ x) \ unknown-id-error} \xrightarrow{Env-unknown-id} \frac{env \vdash (lam \ x \ e1) \Downarrow (clo \ env \ (lam \ x \ e1))}{env \vdash e1 \Downarrow (clo \ env_{old} \ e1)} \xrightarrow{Env-lam} \frac{env \vdash e2 \Downarrow v2}{env \vdash (app \ e1 \ e2) \Downarrow v} \xrightarrow{Env-app} \frac{env \vdash e1 \Downarrow (ds-lam \ x \ e1)}{env \vdash (app \ e1 \ e2) \Downarrow v} \xrightarrow{Env-ds-app} \xrightarrow{Env-ds-app} \xrightarrow{Env-lam \ env \vdash (app \ e1 \ e2) \Downarrow v} \xrightarrow{Env-ds-app} \xrightarrow{Env-lam \ env \vdash (app \ e1 \ e2) \Downarrow v} \xrightarrow{Env-ds-app} \xrightarrow{Env-lam \ env \vdash (app \ e1 \ e2) \Downarrow v} \xrightarrow{Env-ds-app} \xrightarrow{Env-lam \ env \vdash (app \ e1 \ e2) \Downarrow v} \xrightarrow{Env-ds-app} \xrightarrow{Env-lam \ env \vdash (app \ e1 \ e2) \Downarrow v} \xrightarrow{Env-ds-app} \xrightarrow{Env-lam \ env \vdash (app \ e1 \ e2) \Downarrow v} \xrightarrow{Env-lam \ env \vdash (ap$$

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

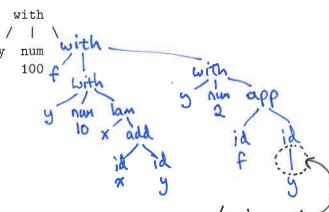
$$lookup((x=(num 2), x=(num 1), \emptyset), x) = (num 2)$$

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

Q1a [10 points] Complete the

abstract syntax tree for the above expression.

(with y (num 100) (with f (with y (num 10) (lam x (add (id x) (id y)))) (with y (num 2) (app (id f) (id y))))



here is optional, but okay