45.

A. Regardless of whether or not $x \in dom\rho$, val creates a new binding

$$\frac{\langle e, \rho\{x \mapsto l\}, \sigma\{l \mapsto unspecified\}\rangle \Downarrow \langle v, \sigma'\rangle}{\langle VAL(x,e), \rho, \sigma\rangle \to \langle \rho\{x \mapsto l\}, \sigma\{l \mapsto v\}\rangle} \quad (DEFINEGLOBAL)\ l \notin dom\sigma$$

B.

```
->(val x 5)
->(val x 6)
->(if (=x 5)
    (println 'NEW-SEMANTICS)
    (println 'OLD-SEMANTICS)
  )
```

My thinking is, if x doesn't equal 5 in the conditional statement then the variable x was updated in its original location (Like it was set) but if x still equals 5 then (val x 6) created a new binding x in a new location. The way location storage and retrieval works in uScheme is unknown to me but it would make sense to me that new variables are stored after previously defined variables and so binding a new x will bind it after the previous x and so retrieval of x would return the first instance of x found in the environment which means that (=x 5) would find the earliest instance of x (5) and not the instance where x=6.

C.

If the new semantics are implemented, and my assumptions from B) are correct, it would make it tremendously hard to know for certain which instance of the variable in question is being recalled. This would make something like (var x (+x1)) impossible, Obviously (set x (+x1)) is the way something like ↑ would be done in the new semantics, but to me this just doesn't seem "complete" I can't see a reason for why any programmer ever would want x=5 and x=6 in the same scope. Both semantics, if you use them correctly will work but the new semantics leave the possibility of multiple "x" variables in the same environment possible, which is unnecessary so I prefer the old semantics where if you happen to accidentally call (var x ...) where x already exists, then x will simply get updated to the new value.