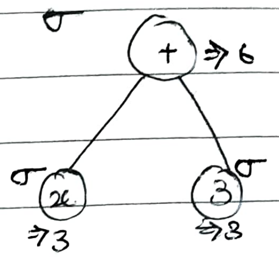
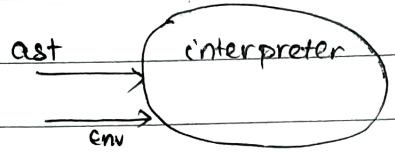
LEXICAL SCOPE LANGUAGE

global vs lexical.





$\sigma = \{x \mapsto 3, y \mapsto 7\}$

LEXICAL SCOPED LANGUAGES

1. abstract syntax
2. evaluator
3. environment
4. parser
5. concrete syntax
6. semantic domains.

way of creating local bindings

```
(let ([x 5] [y 3]) ⇐
  (+ x y))
```

values: expressible,

the result of (returning) an evaluation

denotable.

those ~~that~~ to which an identifier may be bound

storable

values that may be stored in memory.

semantic domains:

expressible values number ∓ boolean

exception vs error-checking.

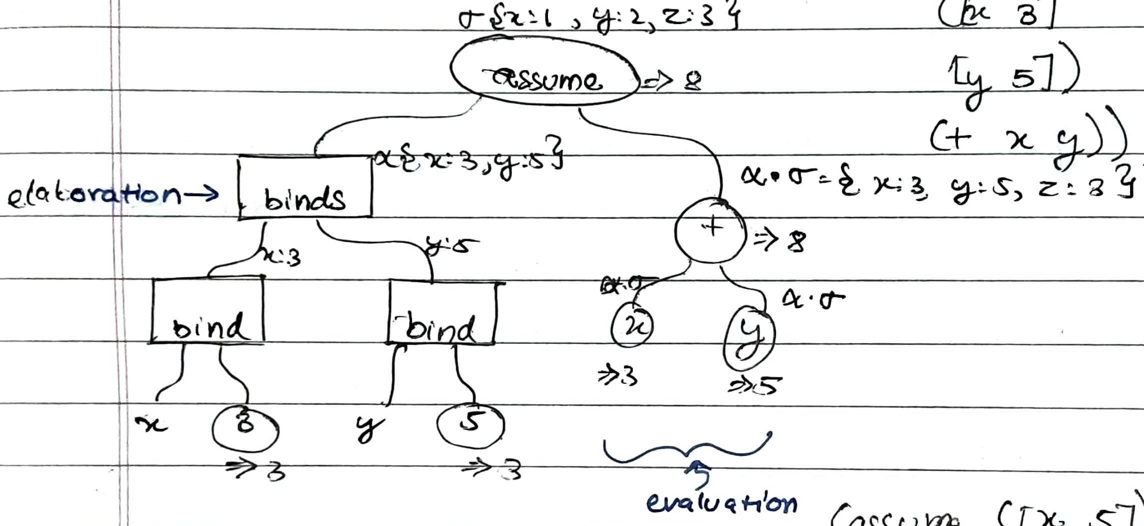
Answer = Expressible Value + Exception
- div by 0
- incorrect types or arguments.
- unbound identifiers.

SYNTAX

```
e ::= n | (op e e) | b
      | x | (assume ([x e] ...) e)

op ::= +, -, *, /
```

```
(assume
  ([x 3]
   [y 5])
  (+ x y))
```



```
(assume ([x 5])
  (assume
    ([y (* 2 x)])
    (+ x y)))
```

denotable values = expressible values.

FUNCTIONS AS VALUES

Semantic domains:

Expressible Values: Number + Boolean + Proc.

```
(let ([f (function (x)  (+ x  1))])
     (f 2))
```

proc = primitive | closure   closure for user defined procedures.

Denotable values = Expressible Values

n | b | f

```
e ::=  n
      | b
      | (assume ([x e]...) e)        local bindings
      | (function (x...) e)
      | (@ e e ...)               (e e...)
      | x                         identifiers.
```

σ = { + : <prim.> , - : <prim>, * : — , / : ... }.