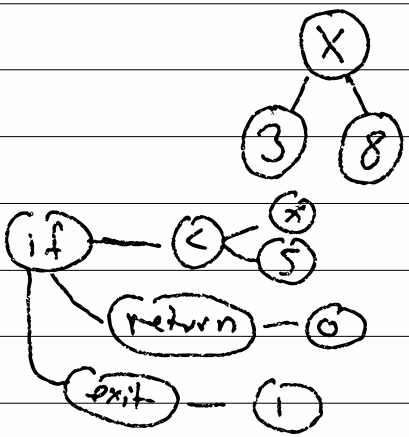
1-1/ 1 + 1 5 1+

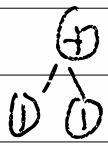
1 x 3 "3 x 8"



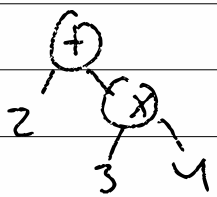

```
if (x < 5) {
    return 0; }
else {
    exit(1); }
```

(+ 1 1)

bp children



(+ 2 (x 3 4))

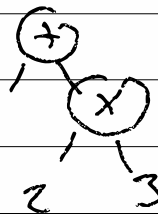


1-2) $J_0 \Rightarrow$ e := v | (+ e e)
v := number | (* e e)

(+ 1 (+ 2 3)) ∈ J_0

```
interface Joe { }
class JNumber implements Joe {
  int n;  JNumber(int _n) { n = _n; }}
class JPlus imp Joe {
  Joe left, right;  JPlus(...) }
class JMult imp Joe {
  Joe l, r;  JMult() ... {} }
```

(+ 1 (* 2 3)) → Sexpr =




```
new JPlus(
  new JNum(1),
  new JMult(
    new JNum(2))
    new JNum(3)))
```

= JP(JN(1), JM(JN(2), JN(3)))

```
class JPlus:
  def _init(l, r):
    this.l = l;
    this.r = r;
```

BST n := mt | (br num n n)

1-3/6 pp : J_0 => string

① pp n = itos (n)

② pp (+ e_L e_R) = "(" ++ pp(e_L) ++ "+" ++ pp(e_R) ++ ")"

③ pp (* e_L e_R) = "(" ++ pp(e_L) ++ "*" ++ pp(e_R) ++ ")"

① interface J0e { public String pp(); }

② class JNum { ...
    public String pp() {
        return intToStr(n); } }

③ class JPlus {
    public String pp() {
        return this.left.pp() ++ "+" + this.right.pp(); }

1-4/ big-step interpreter

```
interp : e -> v

interp n = n
interp (+ eL eR) = interp eL + interp eR
interp (* eL eR) = interp eL * interp eR
```

(1)

```
class JMult {
  public int interp() {
    return this.left.interp() * this.right.interp(); }}
```

```
(+ 1 2 3) = (+ 1 (+ 2 3))
```

desugar →

```
se = empty | (cons se se) | string
(a b c) = (pair "a" (pair "b" (pair "c" mt)))
(+ 1 2) = (p "+" (p "1" (p "2" mt)))
(+ 1 (+ 2 3)) = (p "+" (p "1"
                  (p (p "+" (p "2"
                            (p "3" mt)))
                     mt)))
```

1-5/ desugar for J0

("-" e) => (* -1 (desugar e))

("-" e1 e2) => (+ (d e1) (de ("-" e2)))

("+") => 0

("+" e1 more ...) =>

(+ (d e1) (d ("+" more ...)))

'*'

"x"

(x

= 1

"x"

se => J0 => v

desugar interp

compile bc => v

vm