Algorithms on lists

注意iterator的使用

1. **storing integers**:

Base B
Examples: $B = 100$

72|05

从末尾往前存

$$\begin{tikzcd} \boxed{5} \arrow[r] & \boxed{72} \end{tikzcd}$$

~~35~~6
~~45~~

List stores the "digits" from least significant to most significant.

~~List <Integer> add (List <Integer> a, List <Integer> b) {~~

加法

```
**add** (a, b)        // a, b - lists representing
                      numbers in base B
    carry ← 0
    out ← new List    // output the sum as a list
    it1 ← a.iterator ( )      it2 ← b.iterator ( )
    while it1.hasNext( ) or it2.hasNext( ) or carry > 0 do
        sum ← ~~it1.next( ~~ next (it1) + next(it2) + carry
        out.add (sum % B)
        carry ← sum / B       // int division
    return out
```

```
Helper function next (it):  return it.hasNext ( ) ?
(Failsafe version of next)         it.next ( ) : ~~null~~ 0 ;
```

2. Lists storing sorted sets.

(A1): **Find $L_1 \cap L_2$**

```
intersect (L1, L2, out) :
Helper function next (it) : it.hasNext( ) ? it.next ( ) : null,
    it1 ← L1.iterator( )          it2 ← L2.iterator ( )
    x1 ← next (it1)               x2 ← next (it2)
    while x1 ≠ null and x2 ≠ null :
        if x1 < x2 then            x1 = next(it1)
        else if x1 > x2 then       x2 = next (it2)
        else { out.add (x1)  x1 = next(it1)
                             x2 = next (it2) }
```

哪个小哪个就右移

等于则加入交集，两个都右移

3. Representing sparse polynomials :

表达稀疏多项式

Regular $1 + x^2 - 2x^3 + x^4$ $\quad 1 \to 0 \to 1 \to -2 \to 1$

(coefficient, exponent)
参数 指数

Sparse $3 - 100x^{50} + 97x^{256}$ $\quad (3,0) \to (-100,50) \to (97,256)$

Terms are stored in sorted order of exponent.

代码：
https://stackoverflow.com/questions/28576199/convert-infix-to-postfix-with-stack
http://math.newark.rutgers.edu/~loftin/datafal06/infixToPostfix.pdf

4. stacks: ~~executi~~ evaluation of postfix expressions.

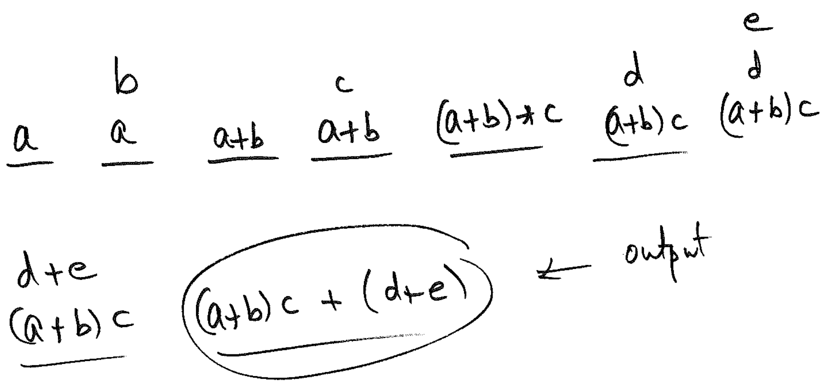
前缀表达式：操作数在运算符之前

a b + c * d e + +

start with an empty stack.
Iterate through postfix expression:
when processing a token:

- operand : Push into stack
- operator : Pop enough items to do operation, perform op, push output into stack

At the end, stack should have one items
↳ result



5. stacks: infix to postfix conversion:

正常顺序转后缀表达式

Ex: a + b * c

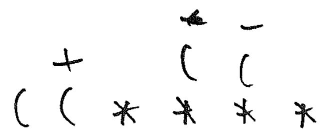
stack of operators
Queue for output.

Iterate over input expression:

- if operand → send to output.
- if operator → Pop enough items from stack and add to output until top of stack has lower precedence than operator. Push operator onto stack.
- (→ Push into stack
-) → Pop from stack and add to output until (is popped.

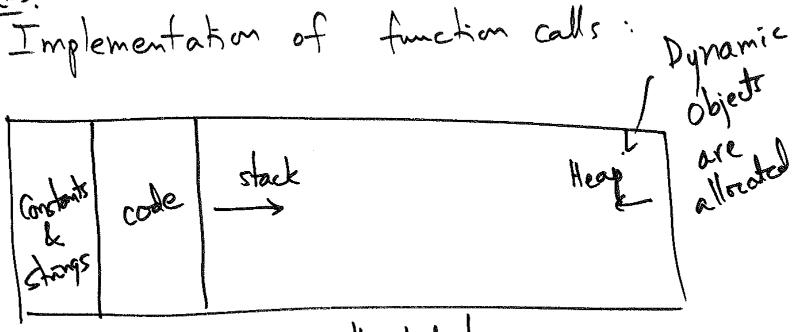
At the end: pop items from stack and add to output until stack is empty.

(a + b) * (c - d)   Queue : a b + c d - *



stacks:

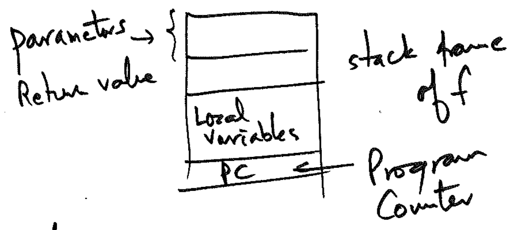
6. Implementation of function calls:



memory allocated to process

Activation Record – stack frame: ~~all~~ space needed to execute one instance of a function.

f(x)



When f( ) is called:

Caller: evaluate actual parameters of call
Pushes into execution stack a frame for f
store caller's PC
Pass control to f.

When f( ) terminates:

f( ):
store return value of f
Restore caller's PC
Pop f's frame from stack.