Syllabus for class Test 1

Unit I
Unit II
} Notes

→ Tomorrow Morning : Assignment (Prototype of model Q's)

Deadline → 22 evening 6 pm.

Arrays → 1D
→ 2D

Matrices → Sparse (Not Dense)
→ Lower Triangular / many matrices
→ RMO / CMO

Stacks
Queues
} array Representation

Applications of stacks

I → Recursion
→ Different types of RC:

- ☆ → Towers of Hanoi

II → Permutations

III → Polish Notation

Q (2M) Dynamic Memory allocation

Permutation

1 2 3

1 3 2

2 1 3

2 3 1

3 1 2

3 2 1

Program

Rec. Tree

Tracing

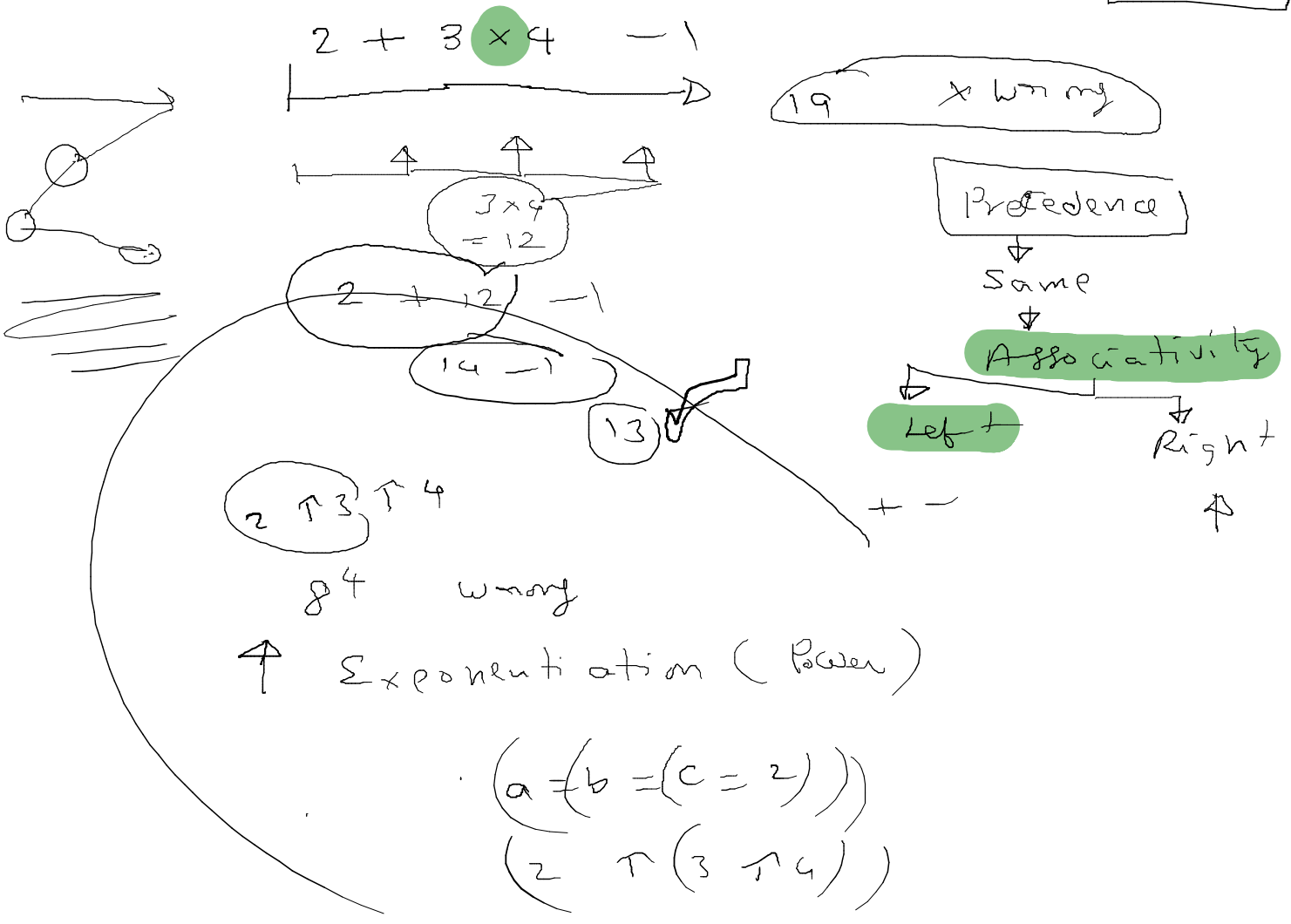
BODMAS

() of Div mul + -

*

+ -

Polish Notation



Refer C / DS → precedence chart

Devised by Lukasiwicz.

→ Polish Logician

→ Avoids repeated scanning of Infix expr

→ In one scan, we get result

{ Prefix / Postfix

Q Convert (log x + b) into prefix / postfix

(−b) " "

(Names) not operator

(a + b) infix

+ ba wrong

+ ab prefix

ba + wrong

ab + ✓ postfix

Note: 1) Relative position of operands: Not disturbed

2) Relative position of operators: disturbed as per precedence and associative Rules

(ex1) (ex2)

a + b * c

Precedence

Pre

Post

a + (b * c)

b * c

* b c

a + (b * c)

b * c

b c *

a +

+ a

+ a * b c

a +

a +

a b c * +

Bottom DFS

pre

Left associative

(* /) : Same level

a * b / c

a * b

* a b

/ c

/ c

/ * a b c

Post

a * b / c

a b *

/ c

c

a b * c /

Unary operator

Infix	$x!_0$	$\log x$	$-b$
pre	$!_0 x$	$\log x$	$- b$
Post	$x !_0$	$x \log$	$b -$

① Try log x !

Assume ! is HP over log

② a = −b + c × d / e ↑ f ↑ g h

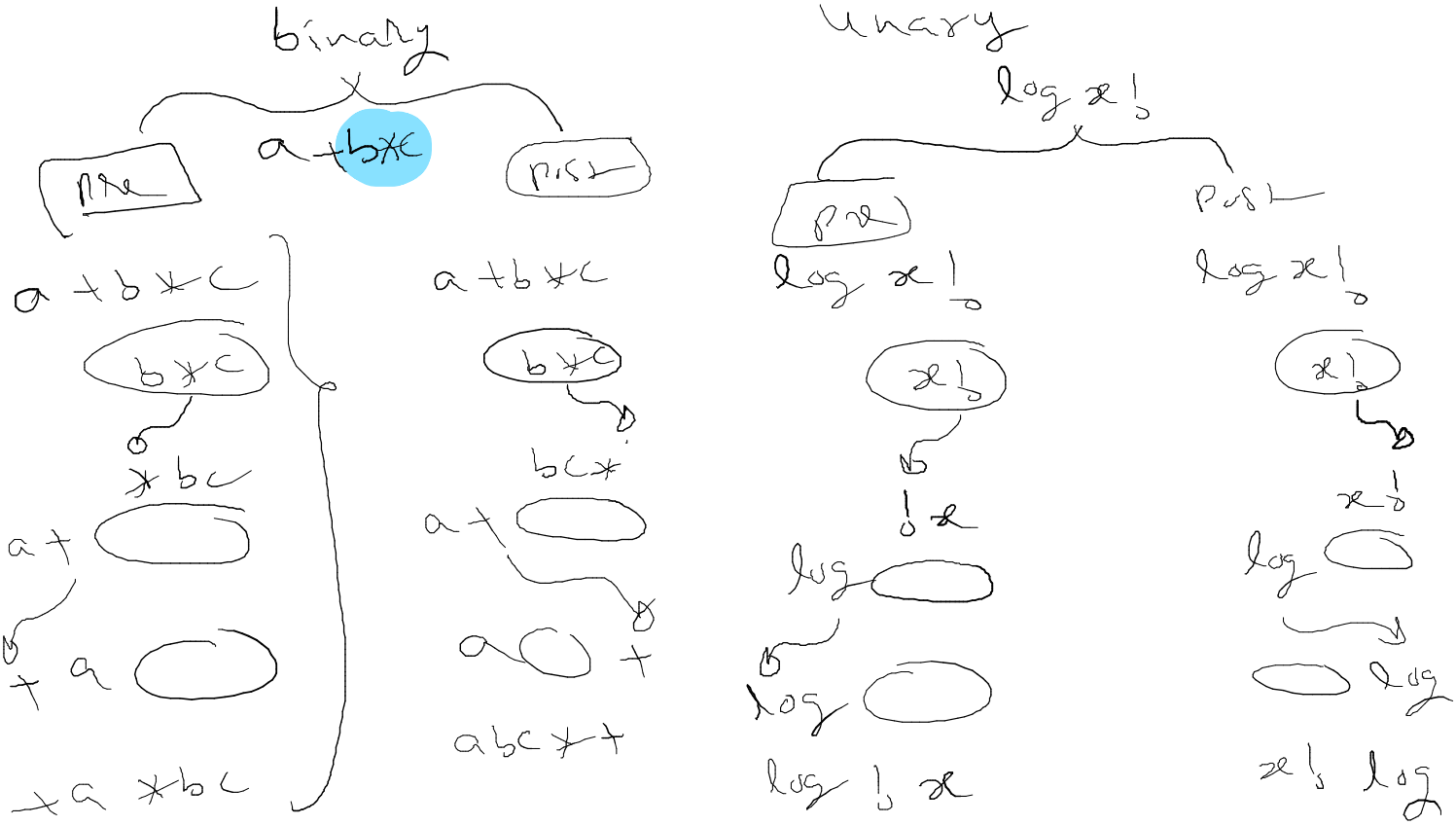
log x !

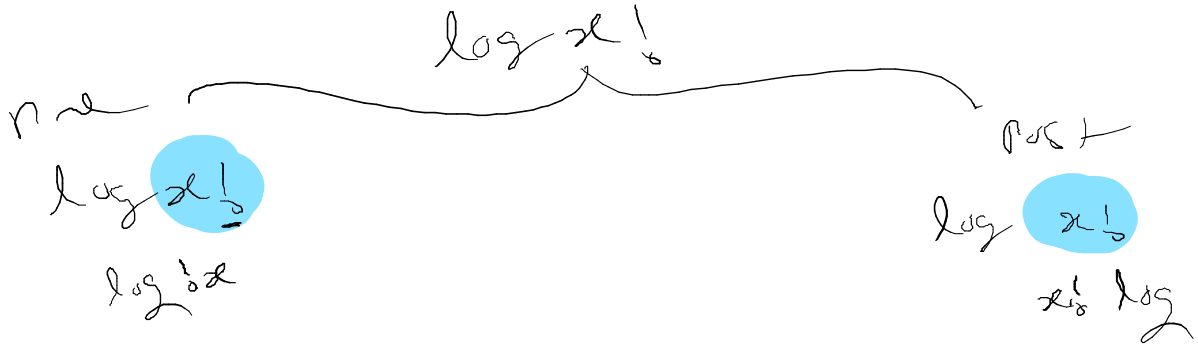
given operator : precedence table

H	!
L	log

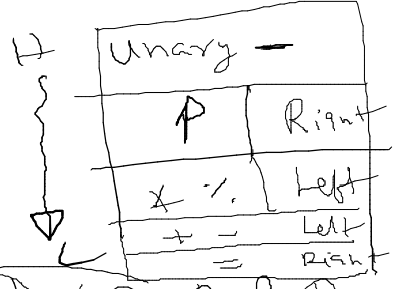
given in the question

Whenever ambiguity, take the help of binary operator example





Given @ operations : Table



✓

PRE

a = –b + c * d / e ↑ f ↑ g – h

–b

–b

f↑g

e ↑ f↑g

c * d / e ↑ f ↑ g

c*d

*cd

*cd↑e↑f↑g

= a – + –b / *cd↑e↑f↑g h

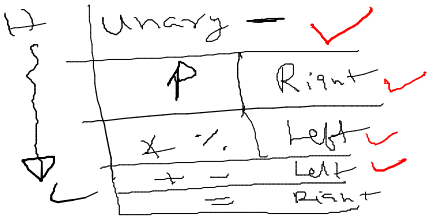
post

a = –b + c * d / e ↑ f ↑ g – h

a b –

c d *

e f g ↑ ↑ / + h – =

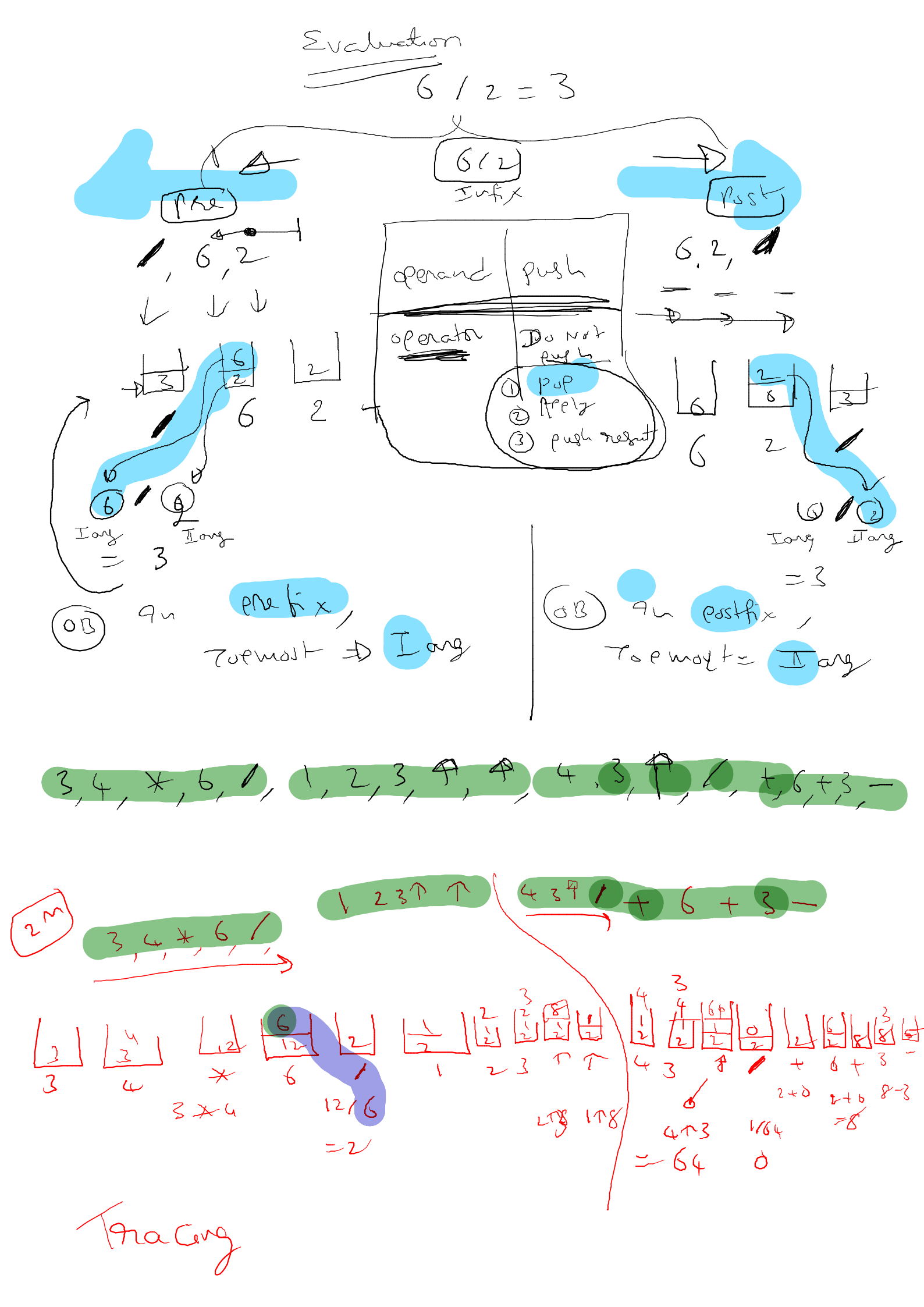


✓ ✓ ✓ ✓ ✓

Try

a << b << c && d > e > f + !g

Break = 5 min

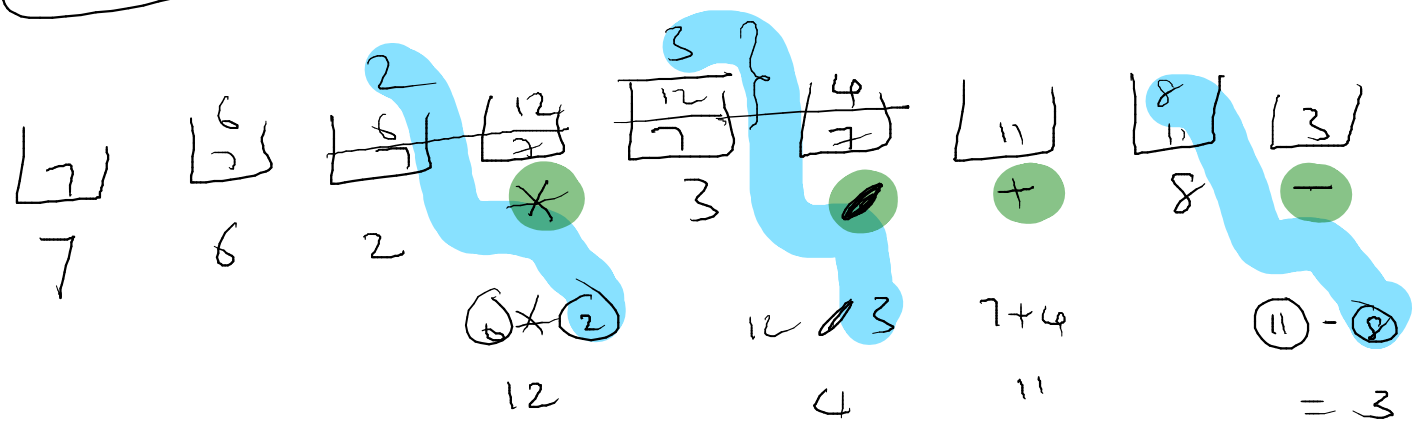

Evaluation
6 / 2 = 3
6/2
Infix
Pre
Post
operand
push
operator
Do not push
pop
apply
push result
Iarg
IIarg
= 3
OB
prefix
Topmost ⇒ Iarg
postfix
Topmost = IIarg
3, 4, *, 6, /
1, 2, 3, ↑, ↑
4, 3, ↑, /, +, 6, +, 3, −
2M
3 * 4
12/6
= 2
4 ↑ 3
= 64
1/64
= 0
2+6
8-3
= 8
Tracing

Practice

Infix 7 + 6 * 2 / 3 − 8 = 3

Post

7, 6, 2, *, 3, /, +, 8, −



Post fix

3 * 4 / 6 + 1 ↑ 2 ↑ 3 / 4 ↑ 3 + 6 − 3

1 2 3 ↑ ↑ 4 3 ↑ / + 6 + 3 −

3, 4, *, 6, /,

Dynamic memory Allocation

→ malloc

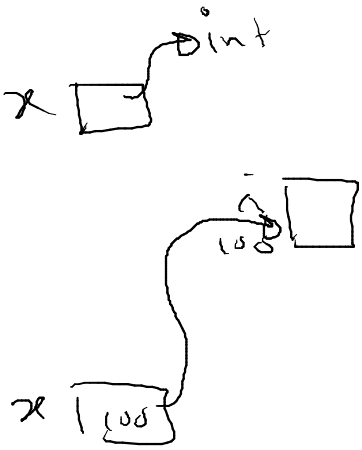
memory allocation

syntax

void * malloc(size)

char * malloc(size)

```
int * x;
int i;
x = &i;
```

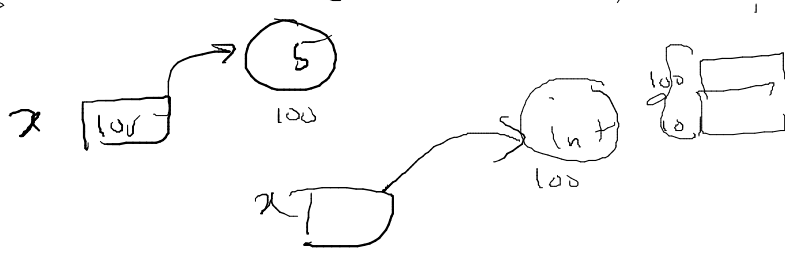


Say ① not given

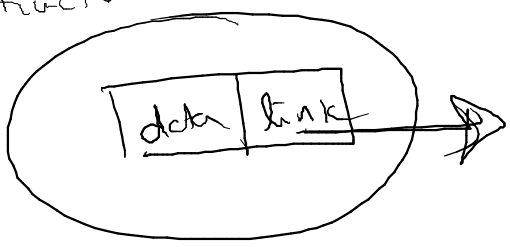
then need to allocate Dynamically

```
int * x;
```


```
x = (int *) malloc(sizeof(int));
*x = 5;
printf("%d", *x);
```



Self Referential structure



node

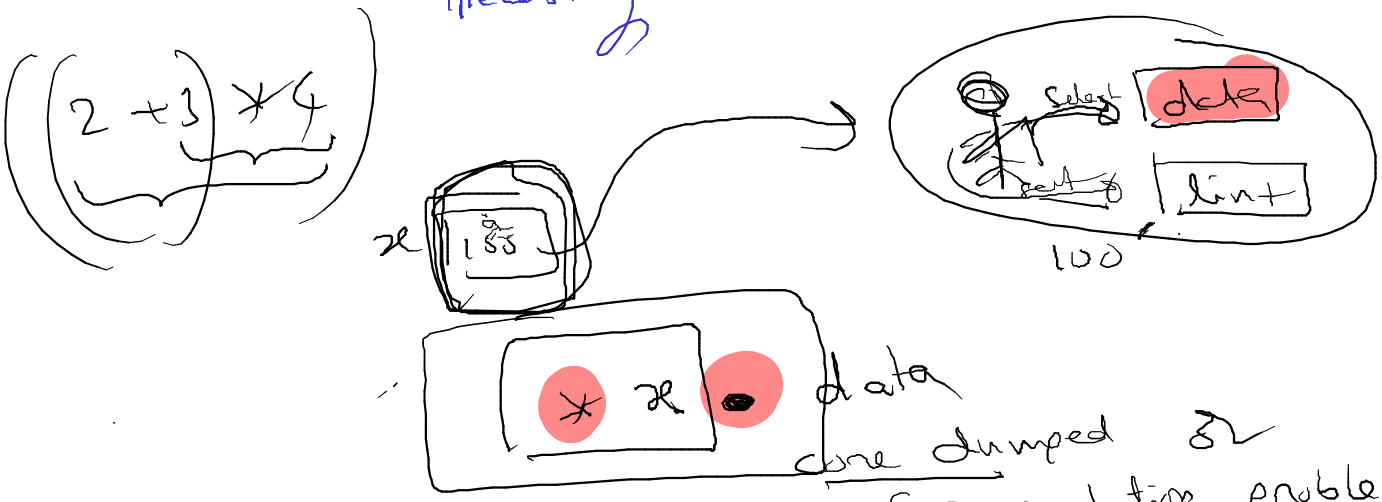
```
struct node
{
    int data;
    struct node * link;
};
```

struct node * x;

x = (struct node *) malloc (sizeof (struct node))

typecasting

((2+3)*4)



core dumped or Segmentation problem

- is higher precedes over *

(* (x . data))

By default * x . data

(* x) . data

x → data

- => direct Selector (m/c friendly)

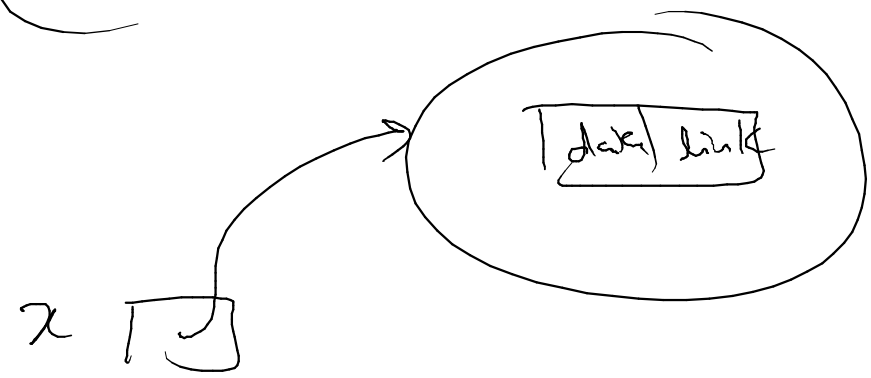
→ Indirect Selector (user friendly operator)

x → data

or

(*x) . data

} Same.

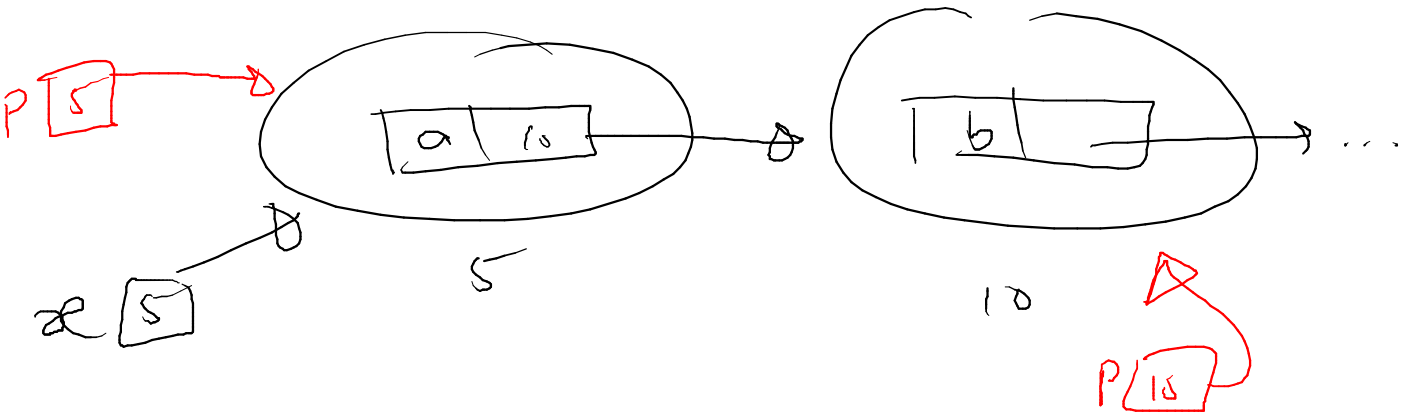


Access

x → data

x → link

move forward



struct node * p = x

p = p → link // forward