EARLY BIRD OFFER

unacademy

FLAT 50% OFF

JOIN THE UMMEED & SANGRAM BATCH FOR GATE DS & AI 2025/26 STARTING ON 9 FEB'24

OFFER LIKE NEVER BEFORE

FOR GATE - CSIT, DSAI & PLACEMENTS SUBSCRIPTION*

	LISTING PRICE	OFFER PRICE	LISTING PRICE	OFFER PRICE
SKU	PLUS	PLUS	ICONIC	ICONIC
12 MONTHS	~~₹39,998~~	₹19,999	~~₹59,998~~	₹29,999
24 MONTHS	~~₹49,998~~	₹24,999	~~₹69,998~~	₹34,999
36 MONTHS	~~₹59,998~~	₹29,999	~~₹79,998~~	₹39,999

Hurry ! Offer valid till **10th FEB**

For more details, Call **85 85 85 85 85**

Use code **CSDA** for maximum discount

*T&C apply, as applicable on the platform

Cyclic Property

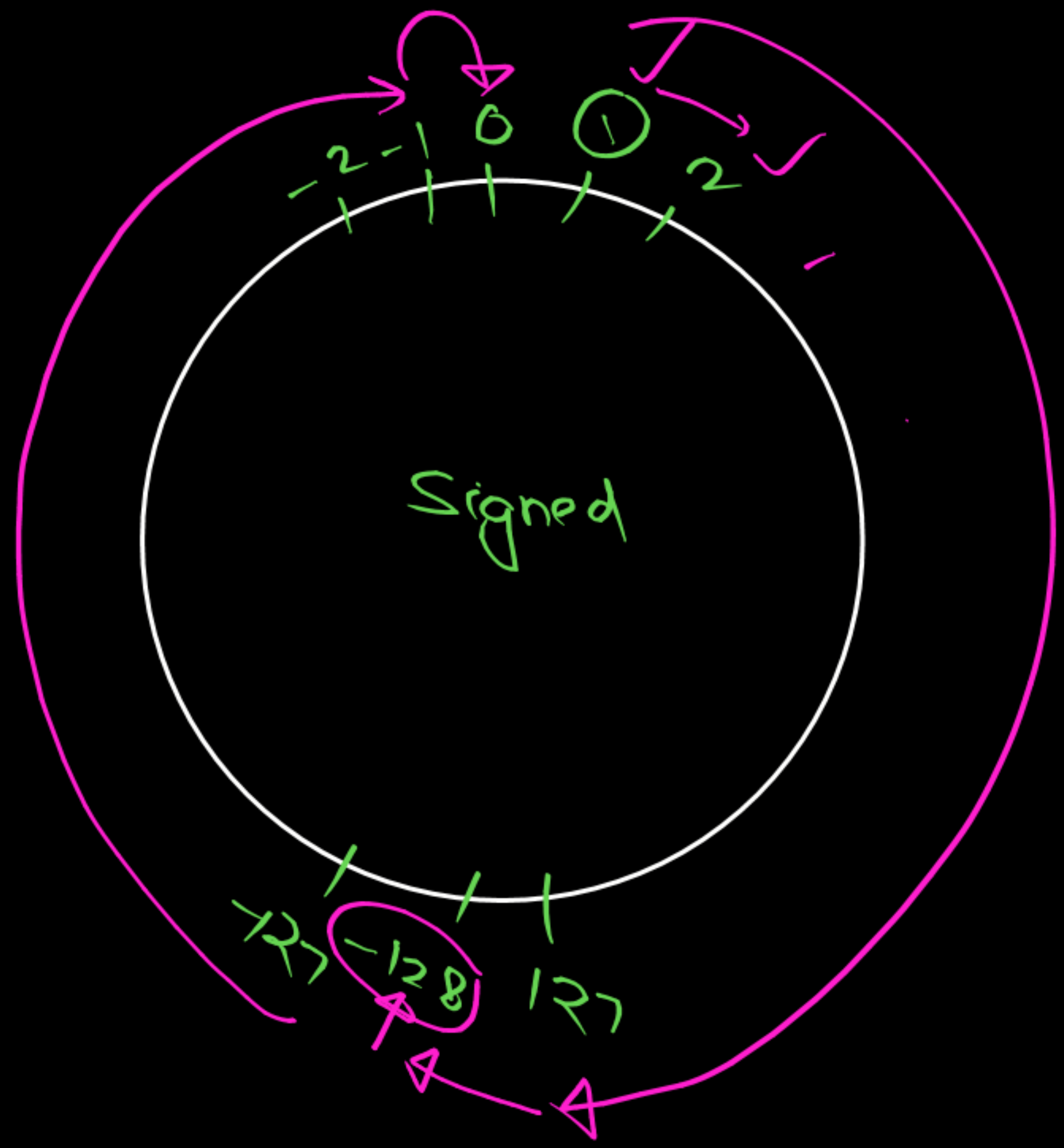
{ char
int
long
long int }

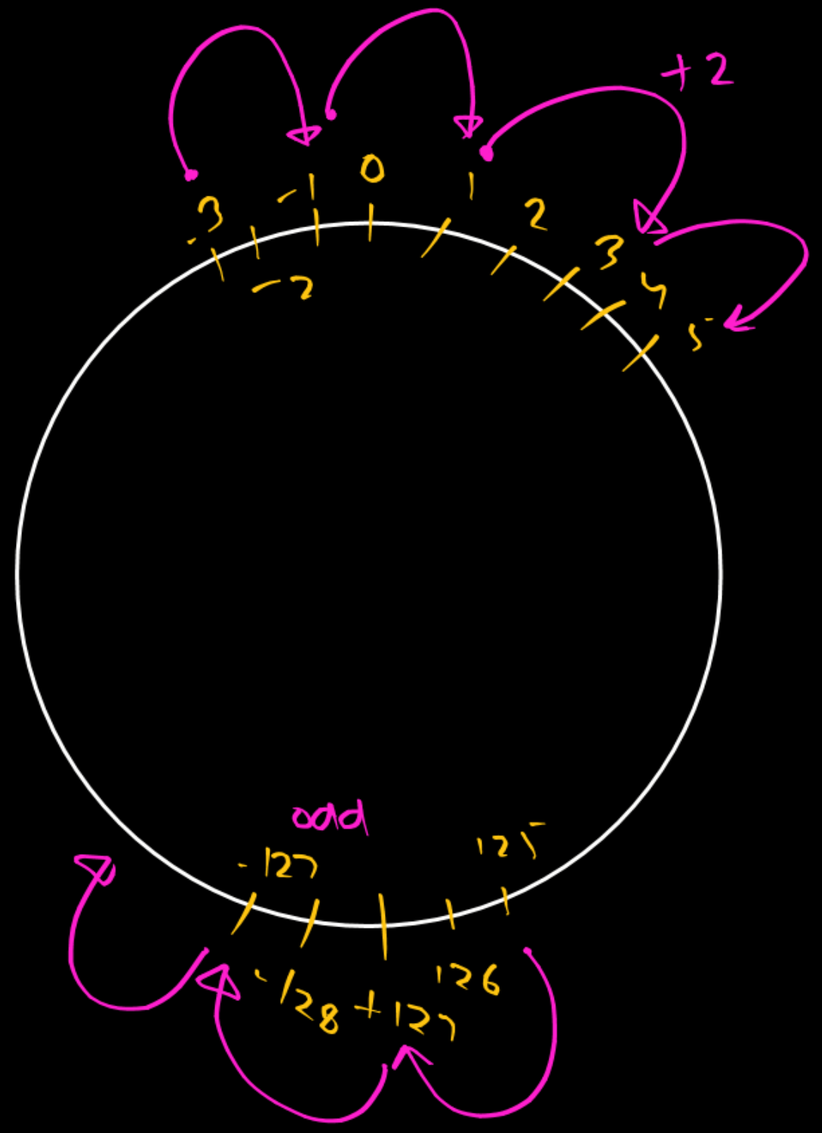
signed char ch = 1;

for (ch = 1 ; ch ; ch = ch+1)

printf("Hi")

255






```
char ch = 1
for ( ch = 1 ; ch ; ch = ch+2 )
        printf("Hi");
```

Operators

Unary	
Arith	x, /, %
	+ -
LS, RS	<< , >>
Rel.	<, <=, >, >=
	!= , ==
bit. and	&
bit XOR	^
bit OR	|
	&&
	||
	?
	=, +=, -= ....
	, (last)

priority

```
int a = ('G' & 'A') | 'T' ^ (('E' && pf("Pankaj")) || pf("cs"));
                 65
print("%d", a);
```

A - 65
B
C
D
E (69)
F
G (71)
H
I
J
K
L
M
N
O
P
Q
R
S
T (84)

71 & 65

64+4+2+1
64 +1

65

a = 65 | 84 ^ ((69 && pf("Pankaj")) || pf("cs"))

True 6

T

T

a = 65 | 84 ^ 1

T || ( ) Eval ✗ Pankaj 85

Switch(exp)

int a = 2, b = 3;

① exp ⇒ Eval ⇒ integer val.

switch('A') ✓

→ 65

switch(12.9) → Compiler ud ke laak

switch(1+2×3)

switch(2 && 3)

switch(printf("Hello"))

switch(2)

switch(a+b×3)

switch(!2)

switch(-3)

✓✓

② Default ⇒ optional
position ⇒ optional

③ break → optional

1st step

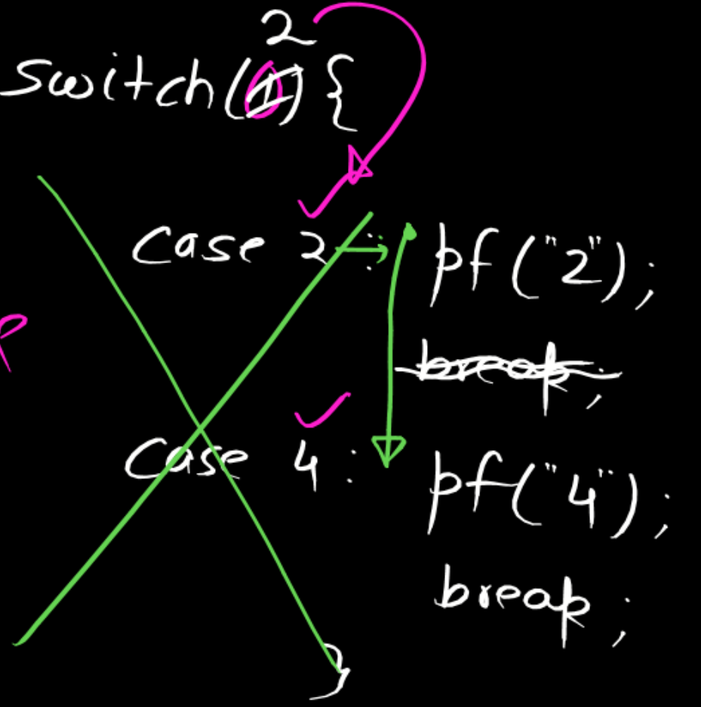
```
switch(2) {
Case 2 : pf("2");
         break;
Case 4 : pf("4");
         break;
}
```

② Default → optional
position → optional

③ break → optional

④ case labels → can not be variable ✗
const / literals ✓

Compiler Ud
ke load




```
int a=2;
switch(1){
Case a : pf(" Mar jaoge");
break;
}
```

24

(5) Duplicate case labels not allowed.

```
switch(1){
    case 65:
        ;
        break;
    case 64+1:
        ;
        break;
}
```

⑥

```
switch(1){
  case 64: Code
           break;
  case 70: code1
           break;
}
```

```
int i = 2;
switch(i){
  i = i+2;        // ignored
  case 2 : pf("2");
           break;
  case 4 : pf("4");
           break;
}
```

(7)

switch(2){ ✓

}

Dummy

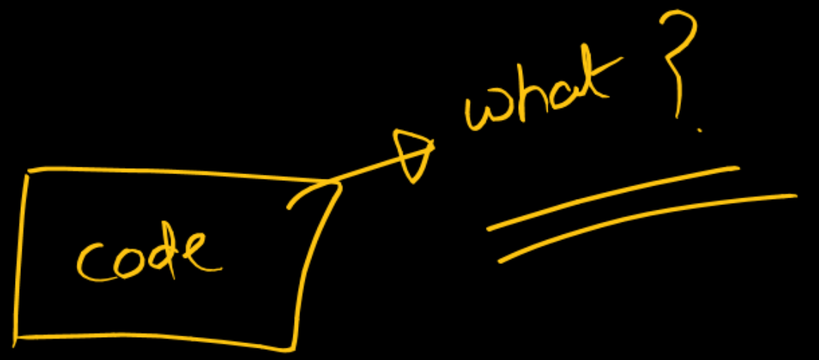
switch(2); ✓

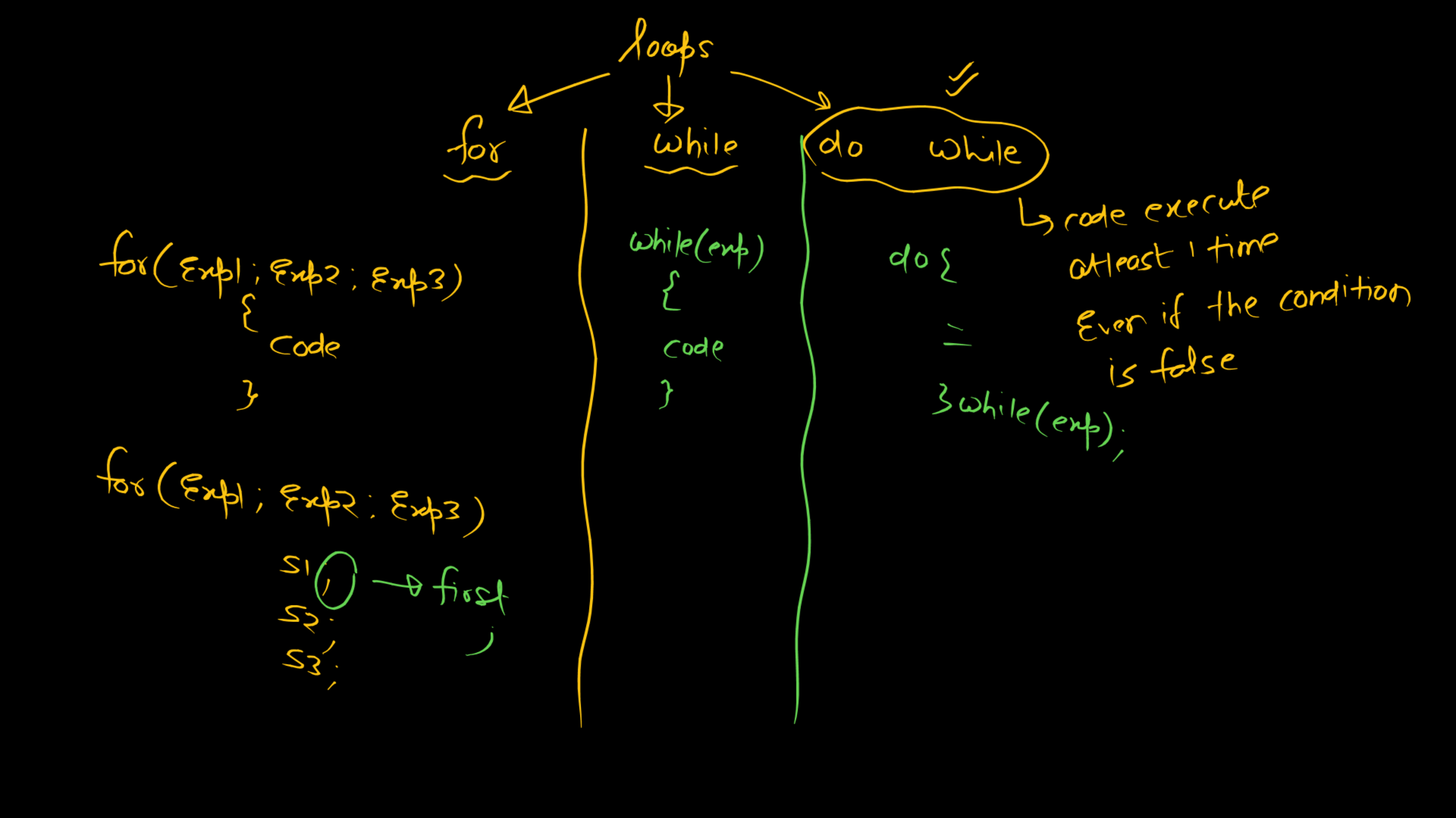
Dummy

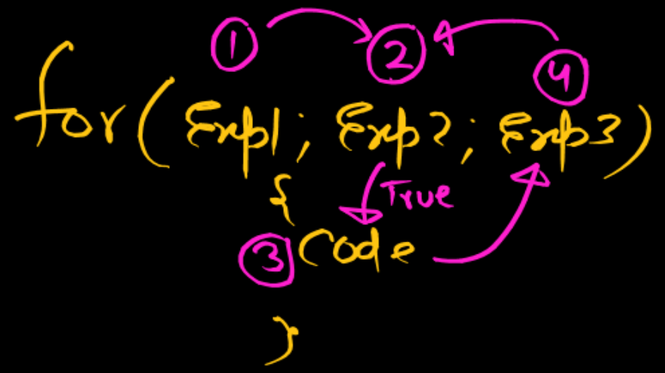
switch(); ✗ → Not Optional

switch(2){

case 2 :




loops
for
while
do while
for(Exp1 ; Exp2 ; Exp3)
{
code
}
for (Exp1 ; Exp2 : Exp3)
S1 ( ) → first
S2 ;
S3 ;
while(exp)
{
code
}
do {
=
} while(exp);
↳ code execute atleast 1 time
Even if the condition is false



Exp 1, 2, 3 ⇒ optional

```
i=1;
for(    ; i<5; i=(i+1))
{
    pf("Hello");
}
```

i		
1	1<5 ✓	pf
2	2<5 ✓	pf
3	3<5 ✓	pf
4	4<5 ✓	pf
5	5<5 ✗	false

```
    i=1;
for( ; i<5;    )
   {
     pf("Hello");
     i=i+1;
   }
```

```
for(    ;    ;    )
   {
     code
   }
```

True

∞ times

continue/break

① continue

35 Tough

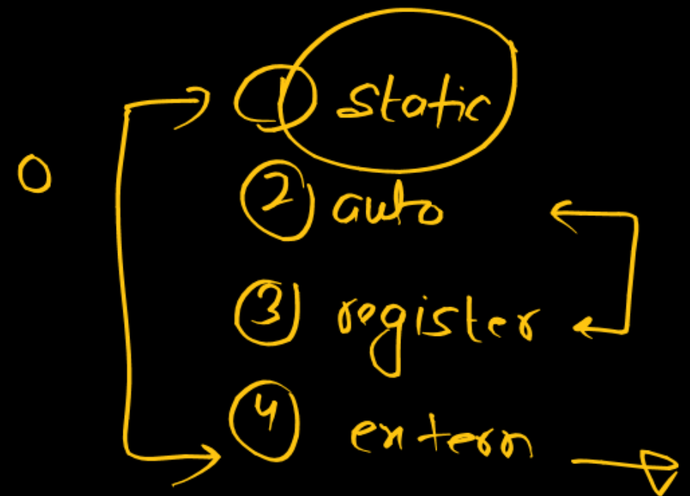
iteration

36

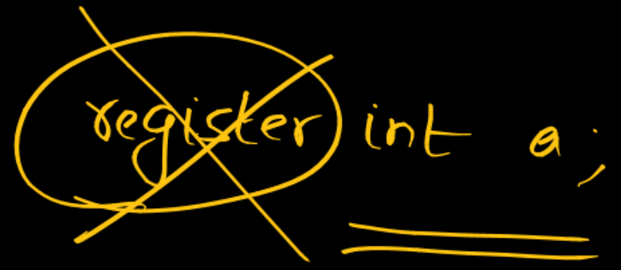
break

34

Storage class



def. value	memory	scope	lifetime
Garbage,	Stack,	block,	block
	register/ Stack		
	data Section		through. program



Arrays / pointers

① Array - name ⇒ constant add. / 1st ele. address

declaration imp.

Ptr + 1

Array - name ++

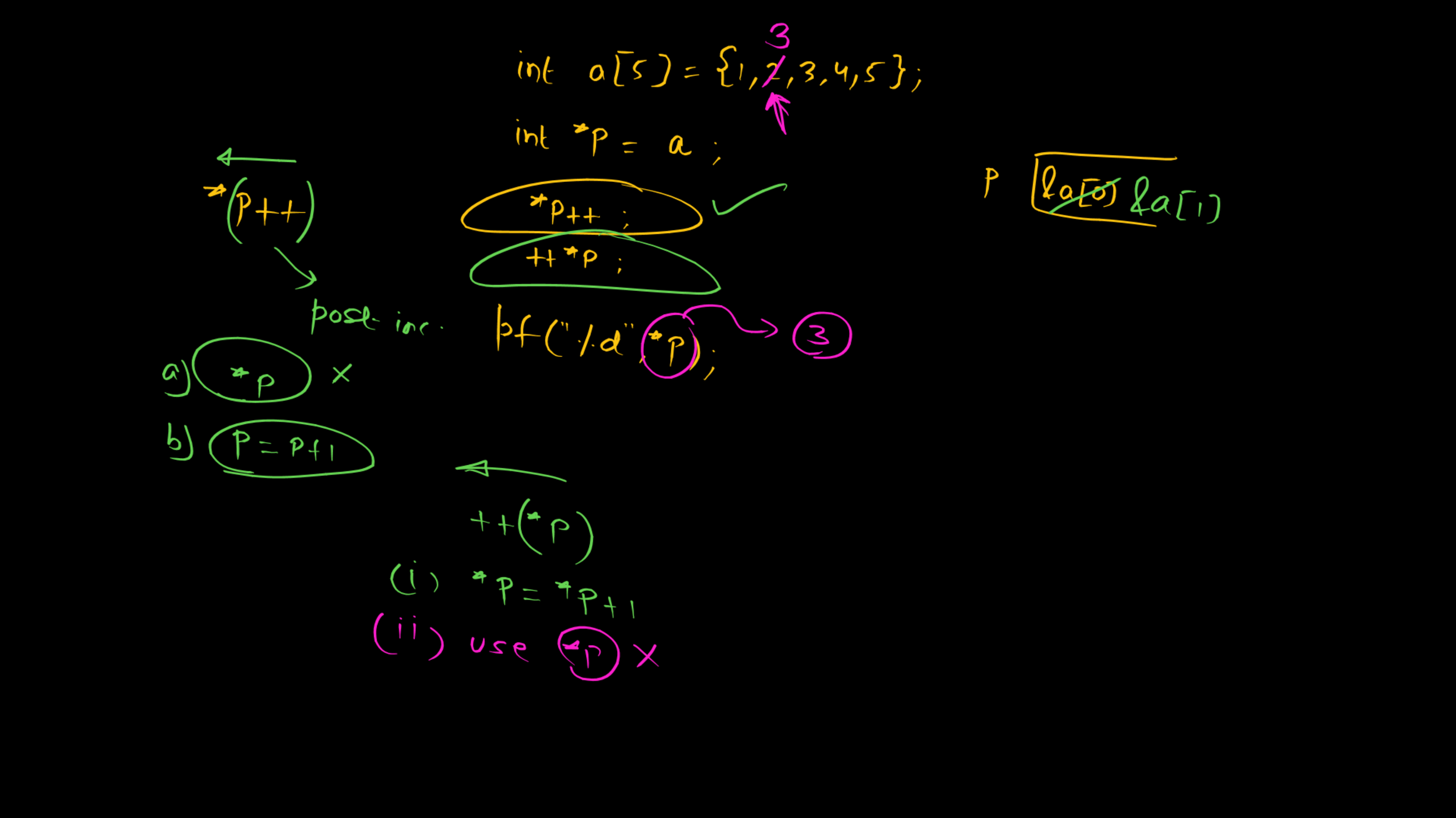
Array - name - -

-- Arr - name

++ Arr - name

Can't be Lvalue

← Arr - name =


int a[5] = {1,2,3,4,5};
3
int *p = a;
*p++ ;
++*p ;
p &a[0] &a[1]
*(p++)
post-inc.
pf("%d", *p);
3
a) *p X
b) p = p+1
++(*p)
(i) *p = *p+1
(ii) use *p X

```
int a[5] = {1,2,3,4,5};

f(a);
bf("%d", a[2]);
```

P

2

3

```
void f(int *p){
    ++*p;
    *++p;
}
```

++(*p)

*(++p)

a) p = p+1

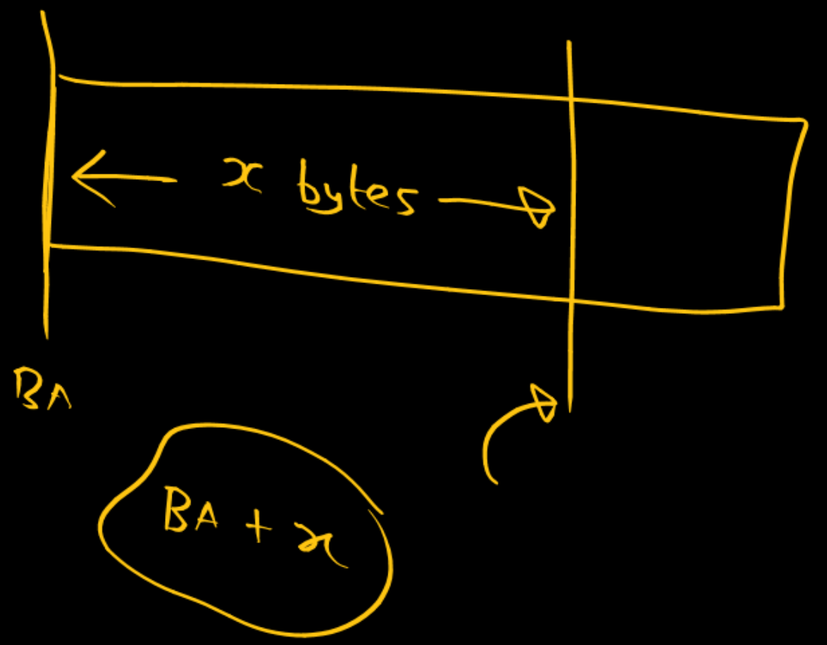
b) *p ×

DS

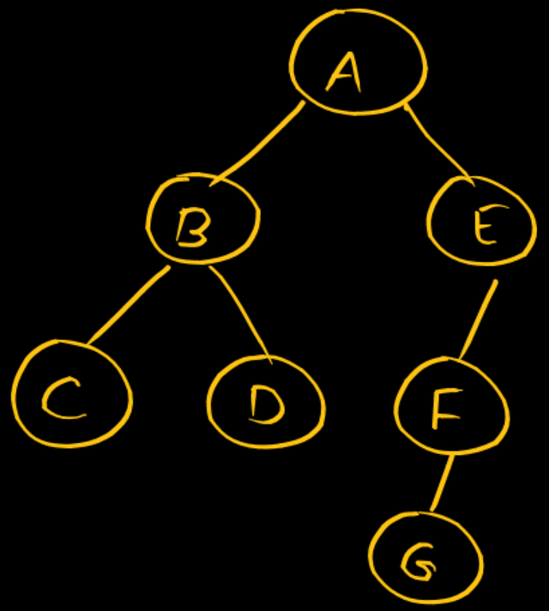
Array → add. calculation

↳ formula → (a) How many ele are filled before target elem

(b) Memory already filled before target element (in bytes)



Trees



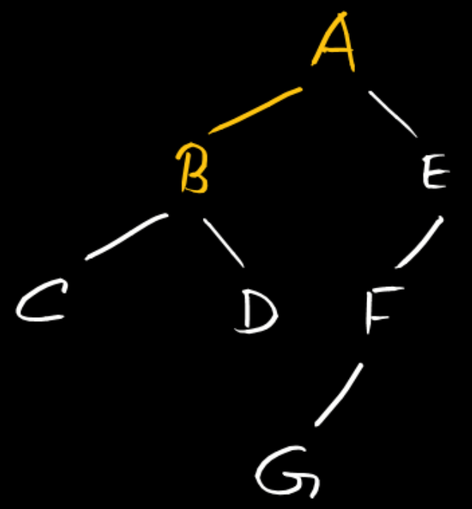
Pre : A B C D E F G

In : C B D A G F E

Post : C D B G F E A

Binary Tree

Trees



Root, L_T, R_T

Pre : A B C D E F G

In : C B D A G F E

→ uniquely

① Pre, In

② Post, In

} Binary Tree

CBT

Either In
or
Pre
or
Post

Why?

FBT

Pre and Post ✓

In/Pre ✓

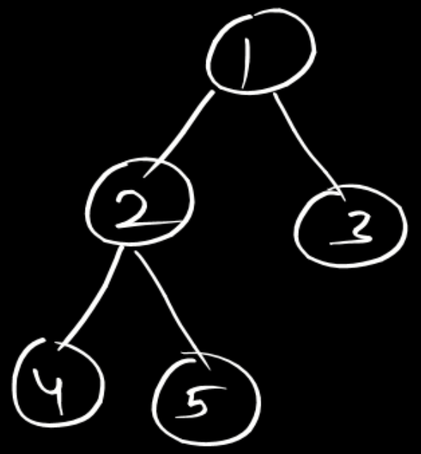
In/Post ✓

0/2 child

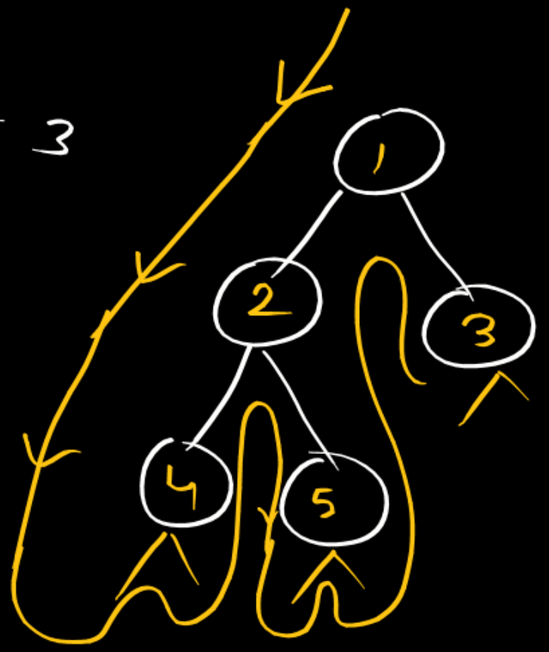
Pre : 1 2 4 5 3

Post : 4 5 2 3 1

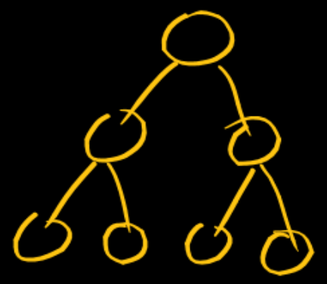
In : 4 2 5 1 3



Pre : 1 2 4 5 3



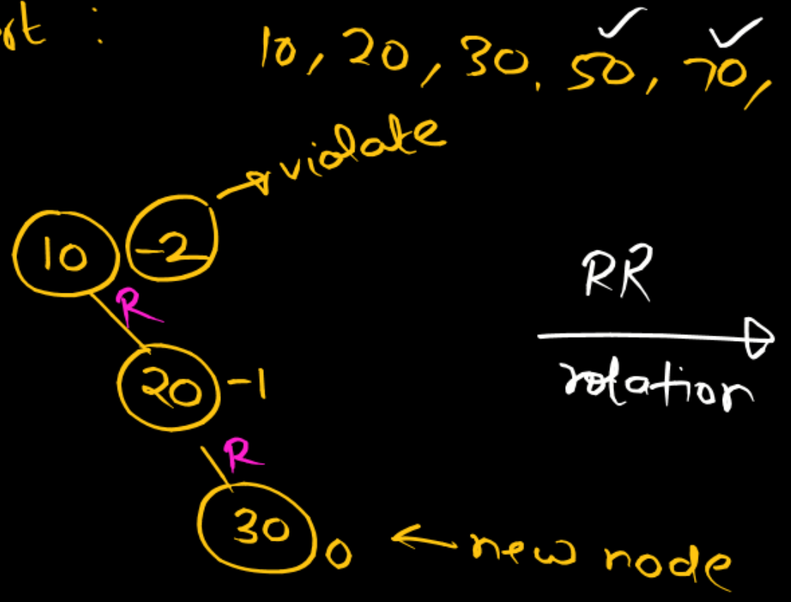
Perfect



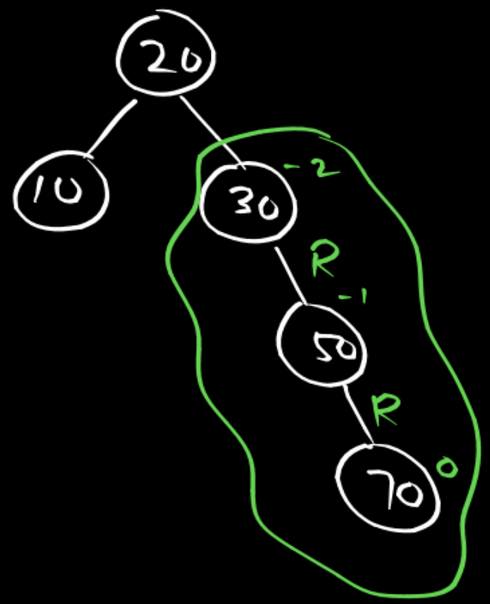
AVL - Tree

{ 0, -1, +1 }

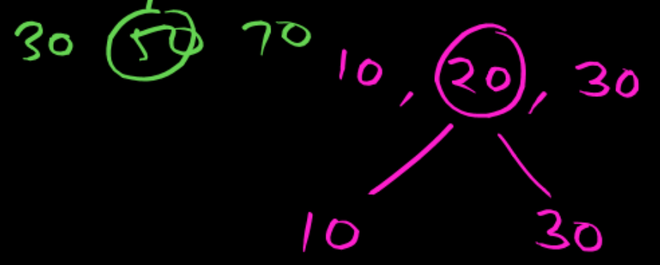
Insert : 10, 20, 30, 50, 70, 35, 40, 46

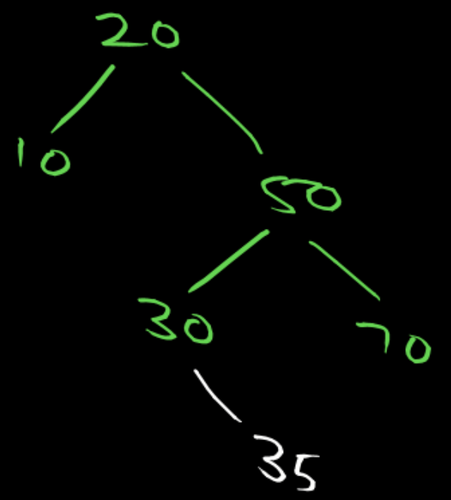


RR rotation



Tri-node structure





AVL - Tree

{ 0, -1, +1 }

Insert : 10, 20, 30, 50, 70, 35, 40, 46

Tri-node structure

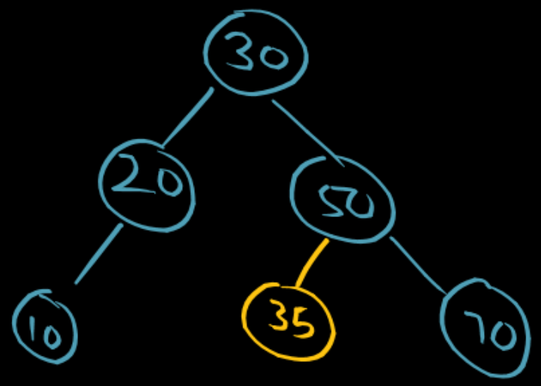
30 50 70

10, 20, 30

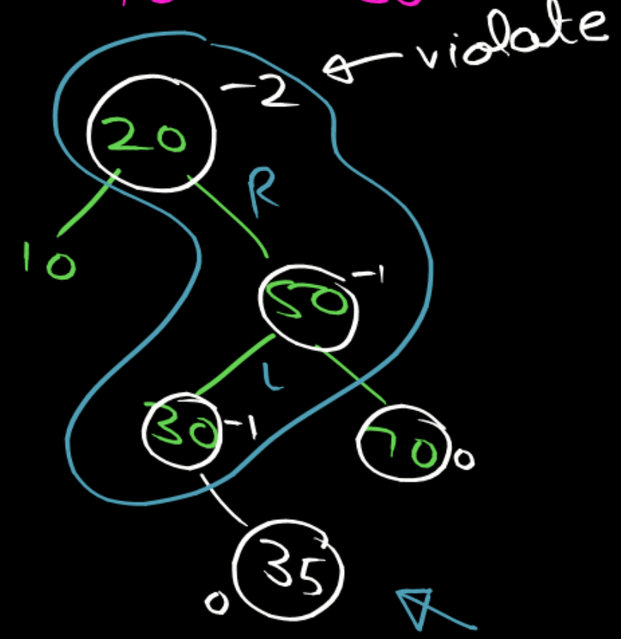
10 30

20 30 50

violate



RL (double)

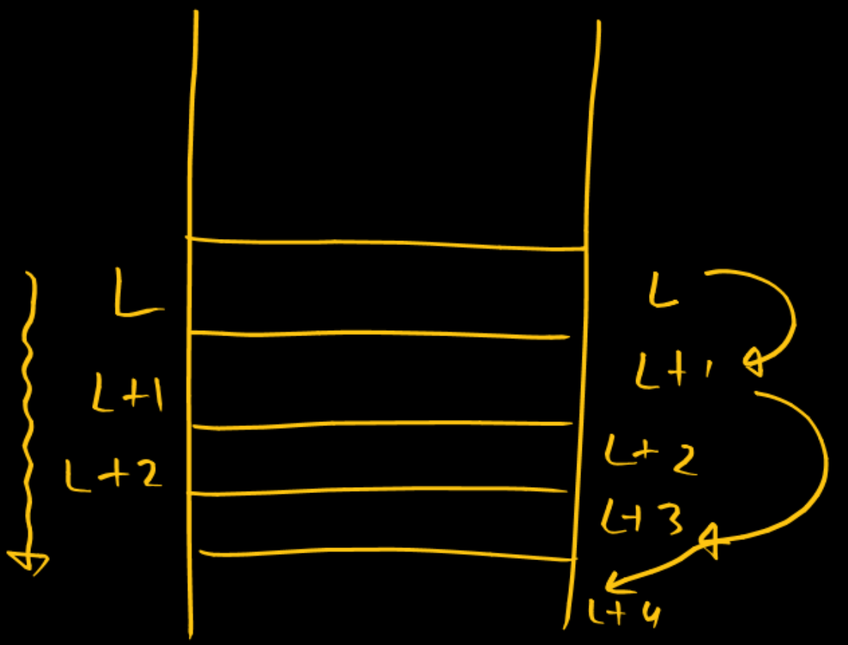


Hashing

Collission

$H(K_1)$ → L

$H(K_2)$ → L



CRT

Prim. clustering

Sec. clustering

Open Addressing

1. Linear Probing → $H(K,i) = (h(K) + i) \bmod m$ (i: coll. No)
   $H(K,1) = (L+1) \bmod m$
   $H(K,2) =$
2. Quadratic Probing → $H(K,i) = (h(K) + i^2) \bmod m$
3. Double Hashing → 2 Hash function → T.C
4. Chaining

CR

Key : 100, 112, 200

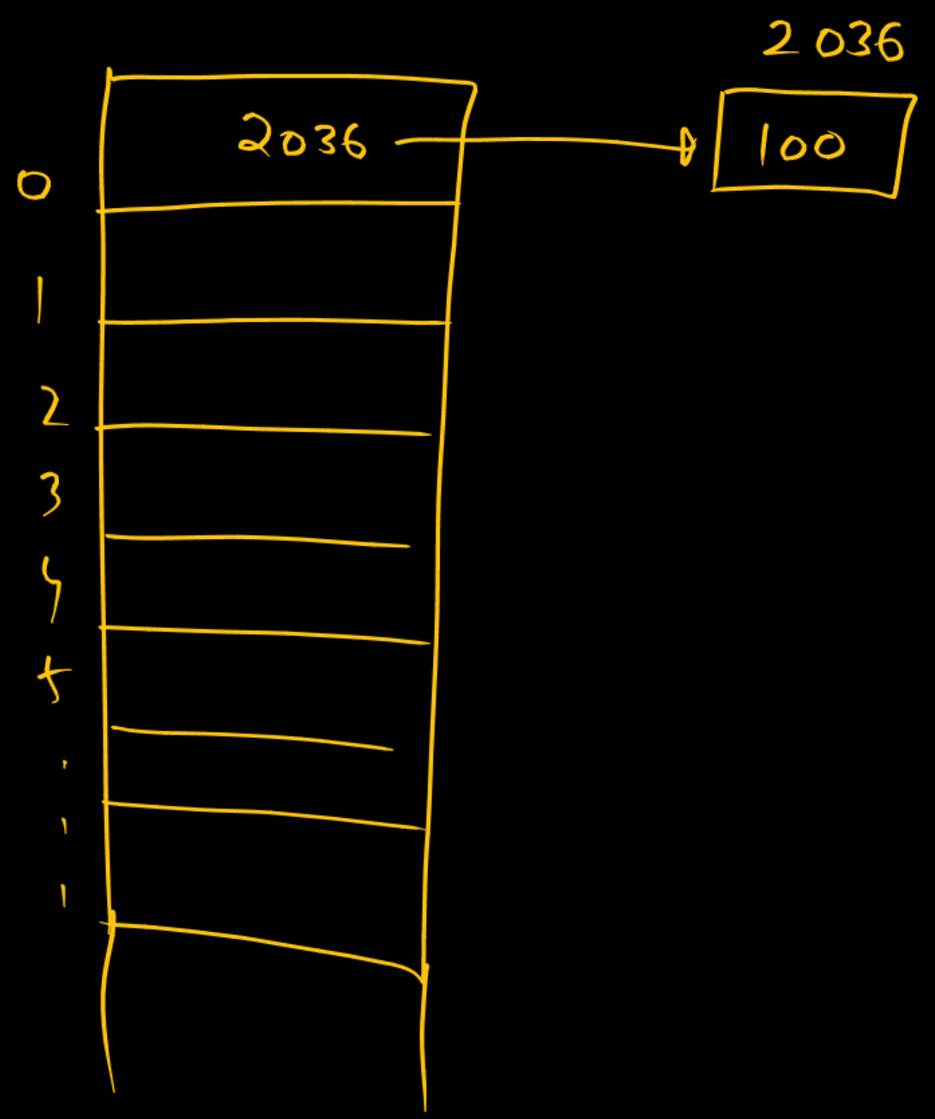
m : 10

$h(k) = 100 \bmod 10$

$= 0$

$h(200) = 200 \bmod 10$

$= 0$



CR

Key : 100, 112, 200

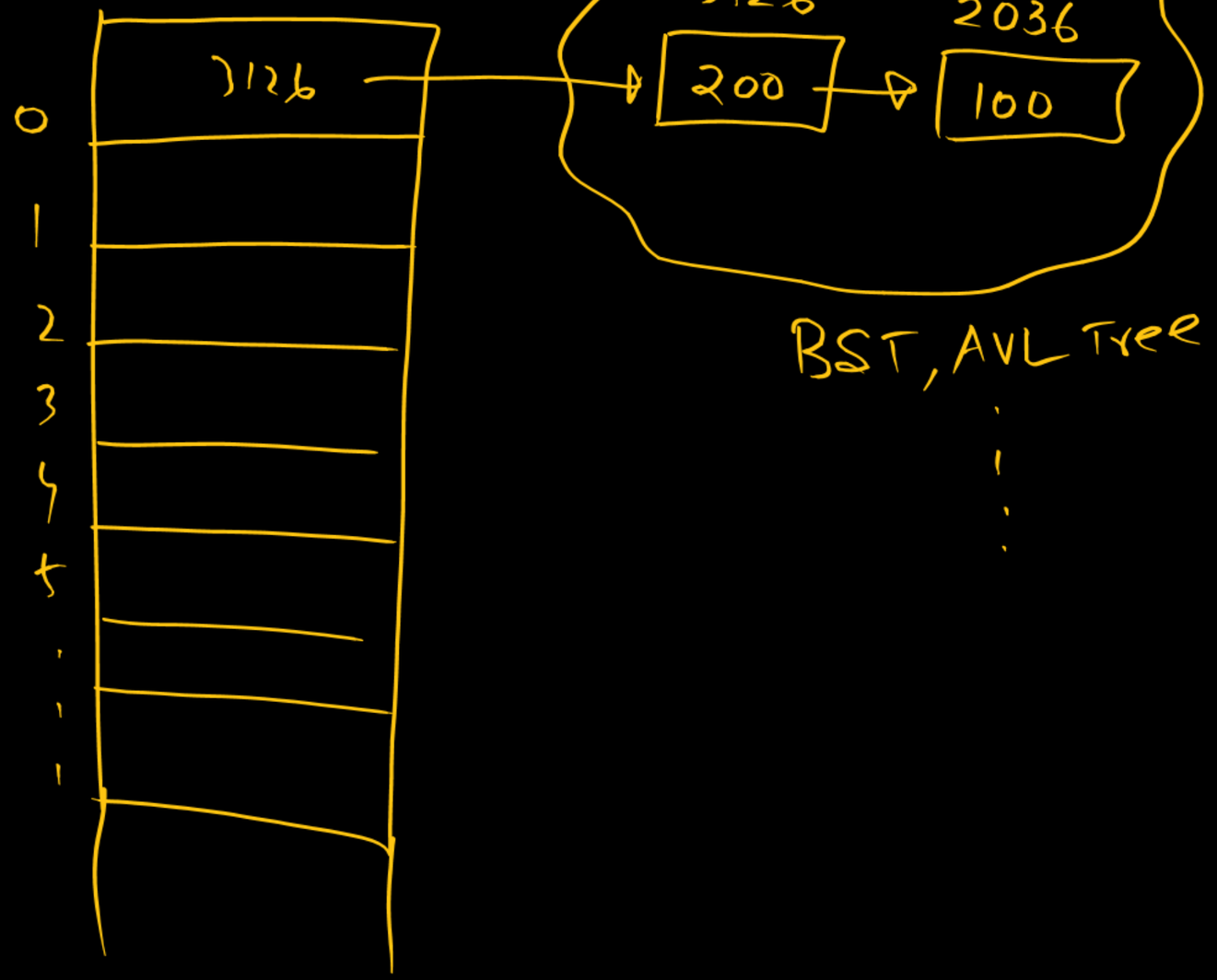
m = 10

$h(k) = 100 \bmod 10$

$= 0$

$h(200) = 200 \bmod 10$

$= 0$



String ✓, strlen, sizeof,

array, pointer, structure

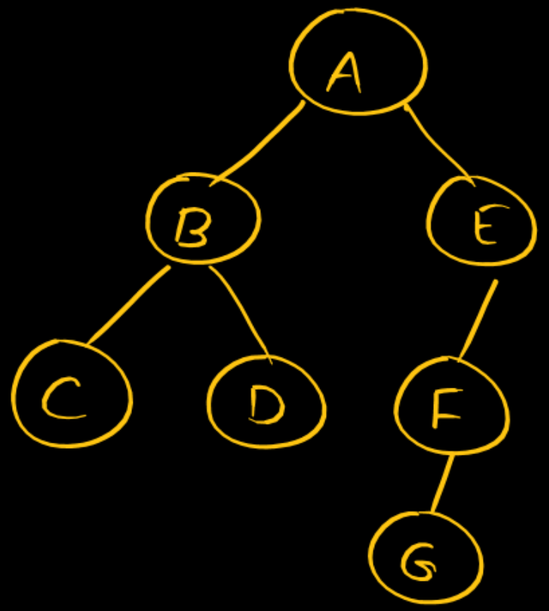
recursion ✓

Trees

Hashing

stack/queue

Trees



Pre: A B C D E F G

In: C B D A G F E

Post: C D B G F E A

Binary Tree

You Can Do It

Believe in Yourself

All The Best!


Thank
you