

Good Eve....

will start by 8.30

Full Stack

Front end

html

css

React

Backend

node.js

Problem Solving

DSA

Java

Python

C++

$\frac{10}{n}$

1 to 10

1 + 2 + 3 + ... + 10

$\frac{n(n+1)}{2}$

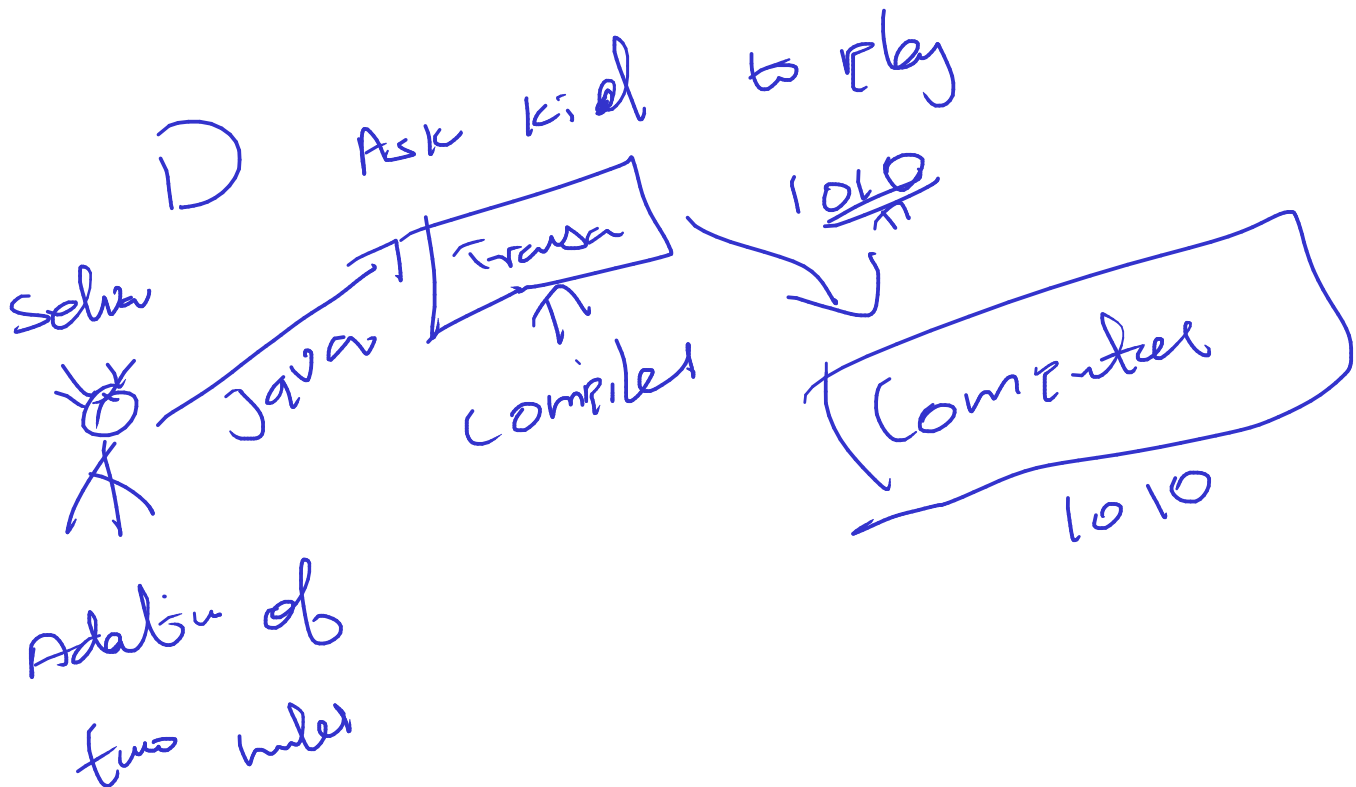
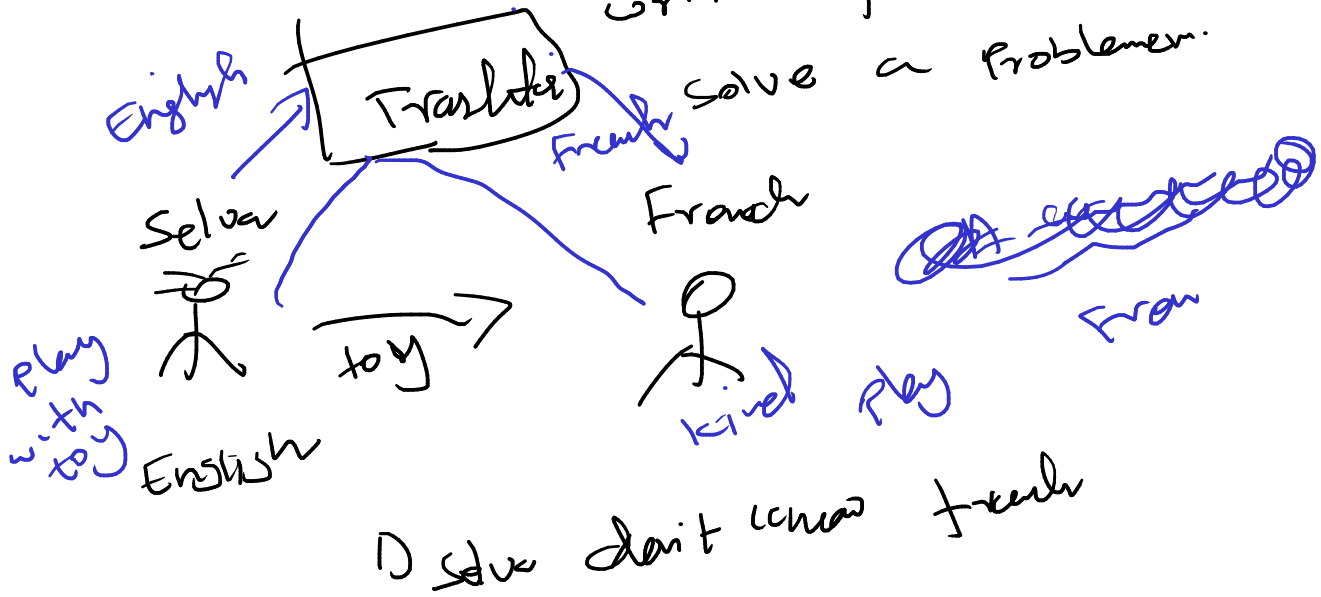
Practice easy

Day 1

easy

# Programming

✓ another set of instructions written for computer to solve a problem.



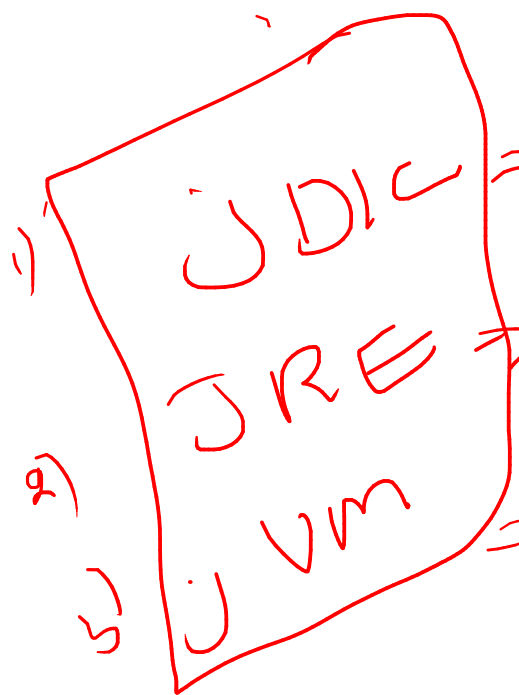
java

IDE X

Eclipse

IntelliJ

vs code



→ java

Development kit

→ java

Runtime Environment

→ java

virtual machine

JDC

Developer

write +

Environment

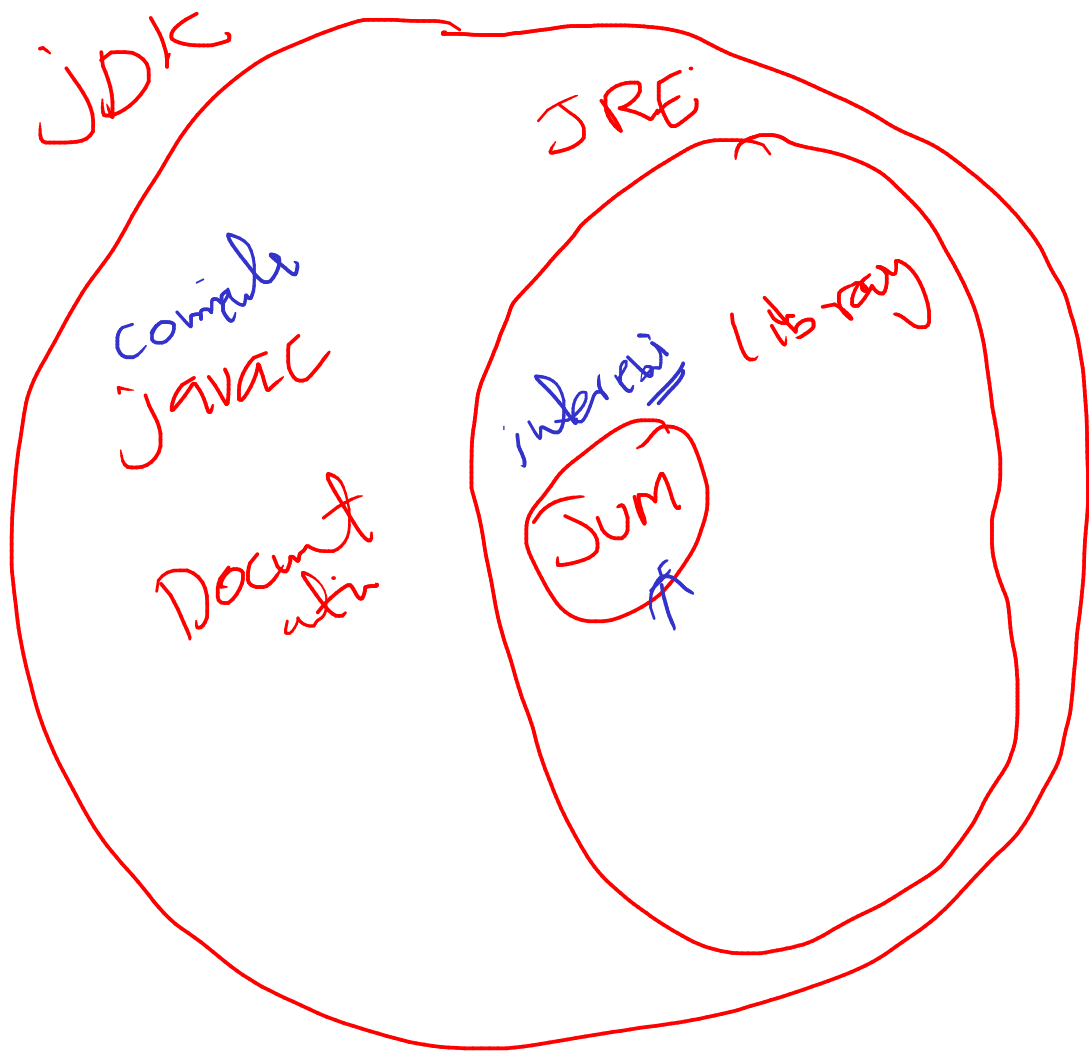
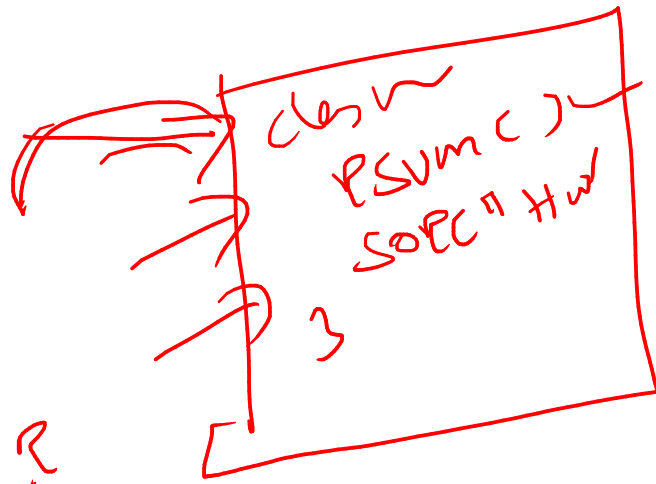
Executing

JRE

Runtime

+ library

giving  
seed line  
by line  
of dr



Drong

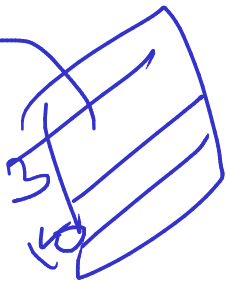
leafw

compile

interest

line by line

all lie at  
one



3 10



(3)

(10)

erro

read

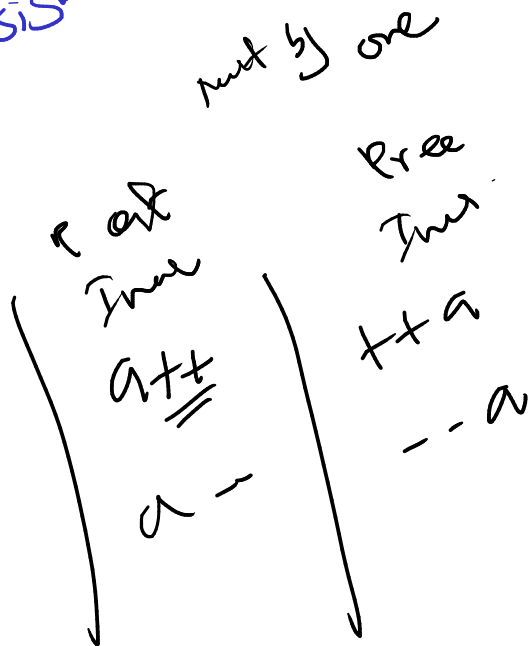
(17)

- 1) inputs
- 2) output
- 3) arithmetic

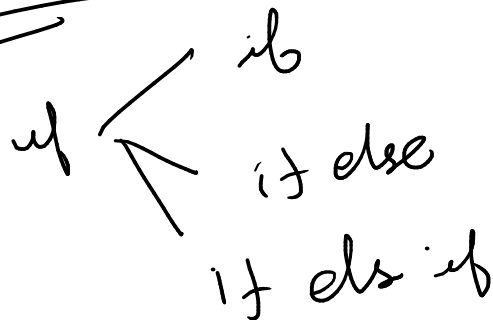
+  
Assignment

algo "

assort  
int a = 10  
a = a + 1



Condition



switch  
conditional ?

Loops

1) HelloWorld

2) Inat    out ent  
      ↓            ↓  
      Scanner    SOP  
      oboutor

+  
-  
/  
\*

conditions

$10 \text{ (==)} 10 \checkmark$

$(10 \neq 20) \checkmark$   
not equal

$(10 == 20) \times$

$10 > 20 \times$

$20 > 10 \checkmark$

$4 < 7 \checkmark$

$10 \leq 10 \checkmark$

number is divisible by 3 and divisible by 10  
yes and  
no

$n = 15$   
 $(15 \% 3 == 0) \checkmark$  T  
 $(15 \% 5 == 0) \checkmark$  T  
yes

And

A	B	o/p
1	0	0
0	1	0
0	0	1
0	0	0

all True  $\rightarrow$  True  
other  $\rightarrow$  False

$(20 \% 3 == 0) \times$  F  
 $(20 \% 5 == 0) \checkmark$  T  
no

number divisible by 3, divisible by 5, divisible by 10

- 1) YES
- 2) NO - False

$30 \rightarrow$   
 $(30 \% 3 == 0) \checkmark$  T  
 $(30 \% 5 == 0) \checkmark$  T  
 $(30 \% 10 == 0) \checkmark$  T  
yes  
no



100 F  
 in  $(100/3 == 0)$  and  $100/5 == 0$  and  
 $(1 == 0)$  X  
 $100/10 == 0$  T

if (E TT) F  
 YES  
 else NO

1) div by 10 ✓  
 2) div by 3 ✓  
 3) div by 5 ✓  
 any of the case YES  
 otherwise NO

100 → 10

25 → T



Add False → False  
 other is True



Logical OR

n=10

F  
 $(10/3 == 0)$  ||  $10/5 == 0$  ||  
 T  
 $(10/10 == 0)$

YES-

Any one is true first YES

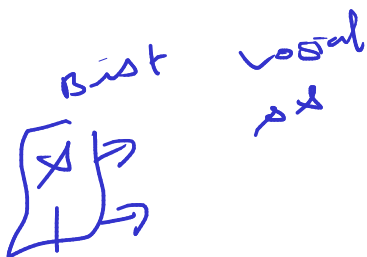
n=77

F  
 $(77/3 == 0)$  ||  $77/5 == 0$  ||  $77/10 == 0$  F

else

NO

OR



Not what | (!)

1  $\rightarrow$  0  
0  $\rightarrow$  1

( $10 \neq 5$ )

if ( $10 > 5$ ) True

if ( $10 < 5$ ) F

!(F) T

False  
!( $10 = 5$ )  
True  
!( $10 > 5$ )  
True  
( $10 \neq 5$ )

True  $\rightarrow$  False !  
False  $\rightarrow$  True !

Loop

if you set of stair  
use your  
Loops

- 1) function
- 1) simple
- 2) return

DType cast

- 2) Assign
- 3) Revision

Function  
 meaningful name } → set of lines of  
 given to code

57 2pm 28th may  
 / Ev: 2pm 30th may

(what / start / what)

irregular mind id . . . 3 coding → 45 mins  
 2) test over go 15mce → 15 ms  
 3) video, 60 ms

1 2 3 4 5 6 . . . Array

'N'

10  
 10.5

int a = 10

double b = 10.5

float c = 10.5f

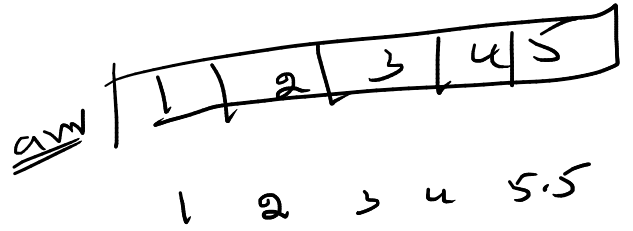
char d = 'N'

10 20 30  
 int e = 20  
 int f = 30

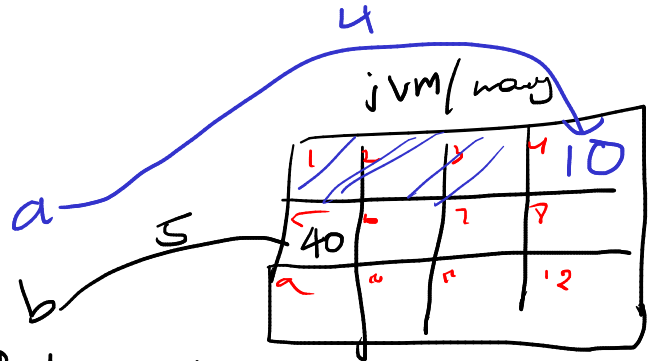
int a = 1  
int b = 2

1 2 3 4 5

z = 26  
mn



int a = 10;  
int b = 40



arr → data type same data / same type data / memory data

int arr

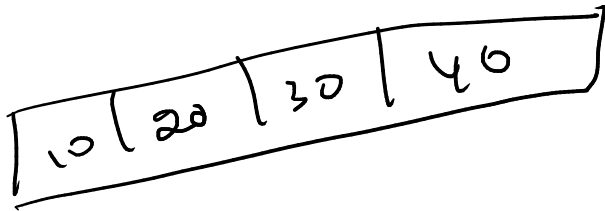
int a = 10  
b = 20

c = 30

d = 40

1 2 3 4 5

arr



int c = 40 A B C D

a = 10  
b = 20

ex

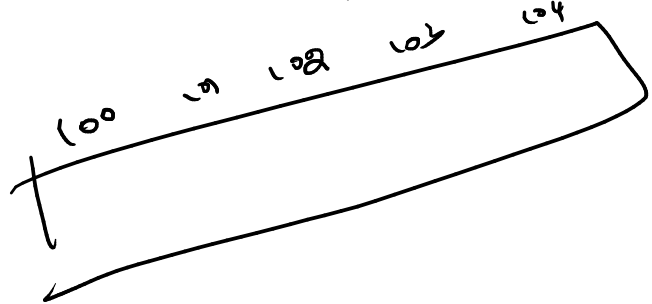
address

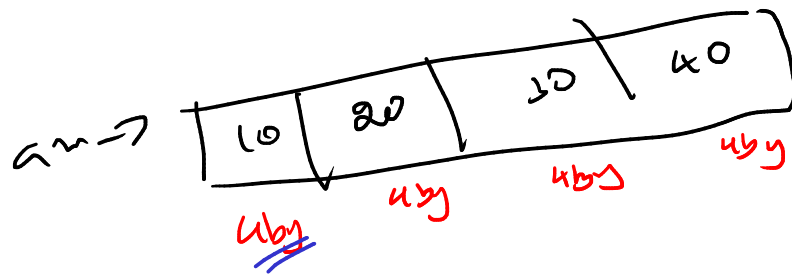
a → 100 = 10

b → 104 = 20

addr  
755 → 40  
758

int  
int





arr → 101

101, 102 103, 104

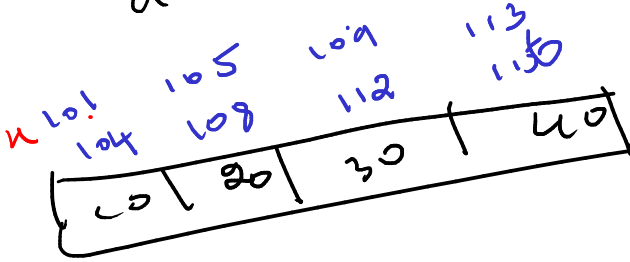
rules system

Binary

Hexadecimal

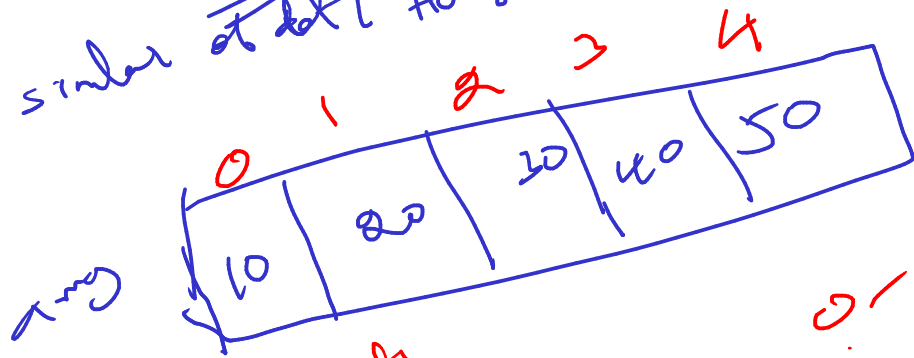
Decimal

Conversion



$4 \times 4 = 16$

similar indexing at all times



size

10 20 30

code

int a = 10  
a = 20  
a = 30

(arr[i])

0 - 4

array  $\rightarrow$  similar type/

int data Homogeneous

11	10.5
----	------

1	2
---	---

10.5	20.5
------	------

distd

1) Array fixed

int arr = new int [5]

0	1	2	3	4
10	20	30	40	50

10 20 30 40 50 60 70 80 arr[5] = 60

1D arr

1D

0	1	2	3
10	20	30	40

arr[0]  
arr[1]

2D  $\rightarrow$  matrix

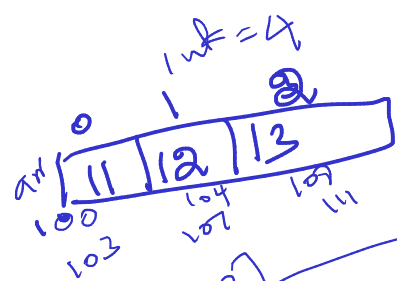
10	20	30	40
----	----	----	----

50	60	70	80
----	----	----	----

20	100	101	102
----	-----	-----	-----

	0	1	2	3
row 0	10	20	30	40
1	50	60	70	80
2	90	100	101	102

arr[Row][Col] = 20  
 $m \rightarrow n$   
m x n



1D

$$100 + 2 \times 4 = 108$$

4 bytes

2D array diagram:

0	1	2	3
10	20	30	40
50	60	70	80
90	100	110	120

Addresses: 100, 104, 108, 112

100 - 147

Row Col

arr[0][1]

$$= 100 + (0 \times (4 \times 4)) + (1 \times 4)$$

$$= 100 + 0 + 4 = 104$$

arr[1][2]

$$= 100 + (1 \times 4 \times 4) + (2 \times 4)$$

$$= 100 + 16 + 8$$

$$= 124$$

1D  $\Rightarrow$  BaseAdd + (size of int \* index)

2D  $\Rightarrow$  BaseAdd + (Row Index \* (size of int \* col)) + (col \* size of int)

# Recursion

→ function calls itself  
until some condition is met  
↓  
Base condition

f()

f() {  
  say hello  
  f();  
}

f() {  
  say hello  
  f();  
}

f() {  
  say hello  
  f();  
}

o/p  
hello  
hello  
hello  
hello  
he

main() {  
  f();  
}

f() {  
  say hello  
  f();  
}

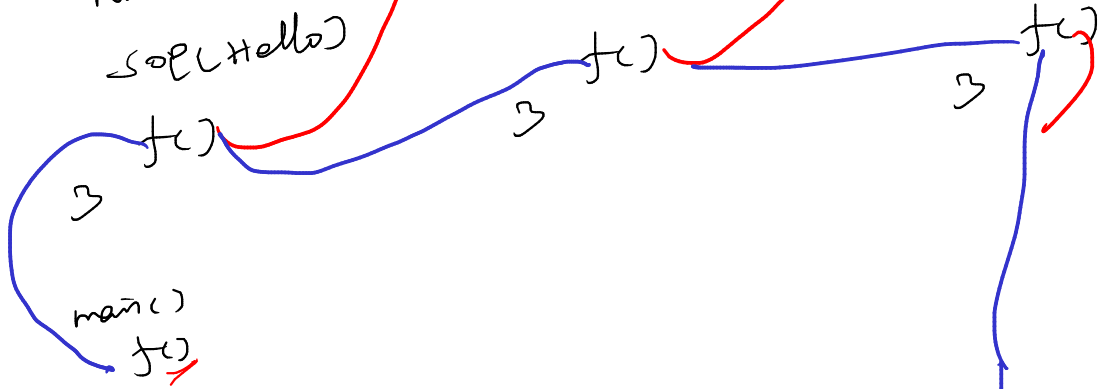


num = 1 2 3 4

f() {  
 if (num > 3) {  
 return;  
 }  
 num = num + 1;  
 sop("Hello");  
}

f() {  
 if (num > 3) {  
 return;  
 }  
 num = num + 1;  
 sop("Hello");  
}

f() {  
 if (num > 3) {  
 return;  
 }  
 num = num + 1;  
 sop("Hello");  
}



f() {  
 if (num > 3) {  
 return;  
 }  
 num = num + 1;  
 sop("Hello");  
}

3 2 1

```
static void f(int i, int n){
    if(i>n){
        return;
    }
    f(i+1, n);
    System.out.print(i+" ");
}
```

```
static void f(int i, int n){
    if(i>n){
        return;
    }
    f(i+1, n);
    System.out.print(i+" ");
}
```

```
static void f(int i, int n){
    if(i>n){
        return;
    }
    f(i+1, n);
    System.out.print(i+" ");
}
```

3  
2  
1

f(1,3)  
f(2,3)  
f(3,3)  
f(4,3)

```
static void f(int i, int n){
    if(i>n){
        return;
    }
    f(i+1, n);
    System.out.print(i+" ");
}
```

f(4,3)  
f(3,3)  
f(2,3)  
f(1,3)

```
static int sumOfNumbers(int i, int n){
    if(i>n){
        return 0;
    }
    int sum = i + sumOfNumbers(i+1, n);
    return sum;
}
```

```
static int sumOfNumbers(int i, int n){
    if(i>n){
        return 0;
    }
    int sum = i + sumOfNumbers(i+1, n);
    return sum;
}
```

```
static int sumOfNumbers(int i, int n){
    if(i>n){
        return 0;
    }
    int sum = i + sumOfNumbers(i+1, n);
    return sum;
}
```

```
static int sumOfNumbers(int i, int n){
    if(i>n){
        return 0;
    }
    int sum = i + sumOfNumbers(i+1, n);
    return sum;
}
```

f(1,3)

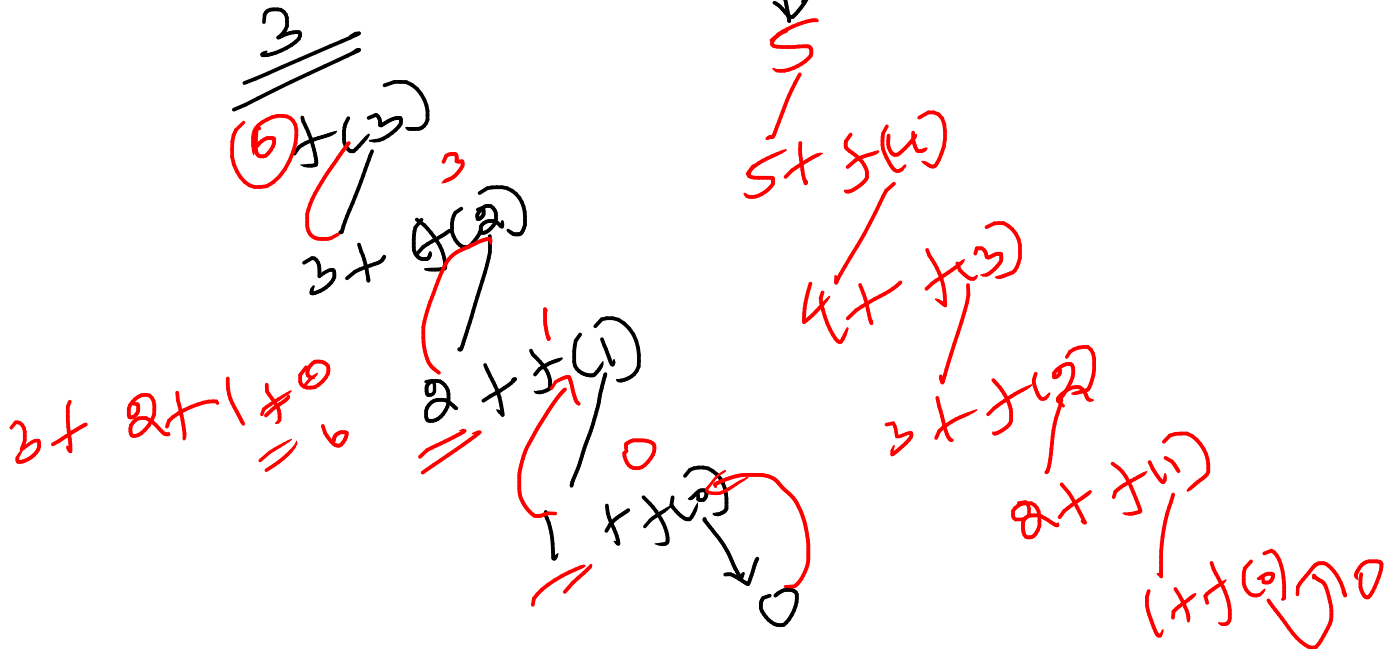


```
static int sumOfNumbers(int n) {
    if (n <= 0) {
        return 0;
    }
    int sum = n + sumOfNumbers(n - 1);
    return sum;
}
```

```
static int sumOfNumbers(int n) {
    if (n <= 0) {
        return 0;
    }
    int sum = n + sumOfNumbers(n - 1);
    return sum;
}
```

```
static int sumOfNumbers(int n) {
    if (n <= 0) {
        return 0;
    }
    int sum = n + sumOfNumbers(n - 1);
    return sum;
}
```

```
static int sumOfNumbers(int n) {
    if (n <= 0) {
        return 0;
    }
    int sum = n + sumOfNumbers(n - 1);
    return sum;
}
```



3!

$3 * 2 * 1 =$

$f(1)$   
 $3 * f(2)$   
 $2 * f(1)$   
 $1$

01-0  
11-1

$\frac{120}{5} \times 5$   
 $\frac{24}{4} \times 4$   
 $\frac{6}{3} \times 3$   
 $\frac{2}{2} \times 2$   
 $\frac{1}{1} \times 1$

RACECAR (1)

1) array ✓  
 2) array ✓  
 3) Review

Review

function  
initial  
method

calling  
some

itself

condition is

stop

```
public static void printHello(){
    System.out.println("Hello world");
    printHello();
}
public static void main(String[] args) {
    System.out.println("Begin");
    printHello();
    System.out.println("End");
}
```

Begin  
H W  
H W  
H W  
H W

```
public static void printHello(){  
    System.out.println("Hello world");  
    printHello();  
}  
public static void main(String[] args) {  
    System.out.println("Begin");  
    printHello();  
    System.out.println("End");  
}
```

```
public static void printHello(){  
    System.out.println("Hello world");  
    printHello();  
}  
public static void main(String[] args) {  
    System.out.println("Begin");  
    printHello();  
    System.out.println("End");  
}
```

```
public static void printHello(){  
    System.out.println("Hello world");  
    printHello();  
}  
public static void main(String[] args) {  
    System.out.println("Begin");  
    printHello();  
    System.out.println("End");  
}
```

```
public static void printHello(){  
    System.out.println("Hello world");  
    printHello();  
}  
public static void main(String[] args) {  
    System.out.println("Begin");  
    printHello();  
    System.out.println("End");  
}
```

~~public static void printHello(){  
 if(count>3){ // condition, base condition  
 return;  
 }  
 count++; // -> count = count+1 increment by 1  
 System.out.println("Hello world");  
 printHello();  
 }~~

~~public static void printHello(){  
 if(count>3){ // condition, base condition  
 return;  
 }  
 count++; // -> count = count+1 increment by 1  
 System.out.println("Hello world");  
 printHello();  
 }~~

~~public static void printHello(){  
 if(count>3){ // condition, base condition  
 return;  
 }  
 count++; // -> count = count+1 increment by 1  
 System.out.println("Hello world");  
 printHello();  
 }~~

public static void printHello(){  
 if(count>3){ // condition, base condition  
 return;  
 }  
 count++; // -> count = count+1 increment by 1  
 System.out.println("Hello world");  
 printHello();  
 }

~~public static void printHello(){  
 if(count>3){ // condition, base condition  
 return;  
 }  
 count++; // -> count = count+1 increment by 1  
 System.out.println("Hello world");  
 printHello();  
 }~~

~~public static void print1toN(int curr, int n){  
 if(curr>n) { //count>3  
 return;  
 }  
 System.out.println(curr);  
 curr++;  
 print1toN(curr, n);  
 }~~

~~public static void print1toN(int curr, int n){  
 if(curr>n) { //count>3  
 return;  
 }  
 System.out.println(curr);  
 curr++;  
 print1toN(curr, n);  
 }~~

~~public static void print1toN(int curr, int n){  
 if(curr>n) { //count>3  
 return;  
 }  
 System.out.println(curr);  
 curr++;  
 print1toN(curr, n);  
 }~~

~~public static void print1toN(int curr, int n){  
 if(curr>n) { //count>3  
 return;  
 }  
 System.out.println(curr);  
 curr++;  
 print1toN(curr, n);  
 }~~



```
public static void print1toN(int curr, int n) {
    if (curr > n) {
        return;
    }
    System.out.println(curr);
    curr++;
    print1toN(curr, n);
}
```

```
public static void print1toN(int curr, int n) {
    if (curr > n) {
        return;
    }
    System.out.println(curr);
    curr++;
    print1toN(curr, n);
}
```

```
public static void print1toN(int curr, int n) {
    if (curr > n) {
        return;
    }
    System.out.println(curr);
    curr++;
    print1toN(curr, n);
}
```

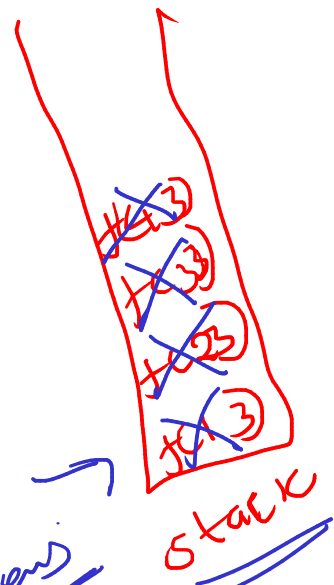
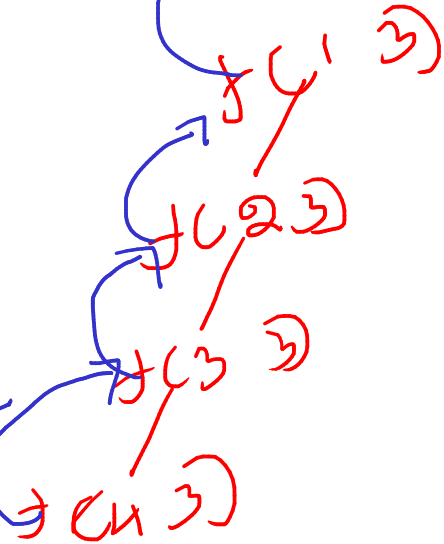
```
public static void print1toN(int curr, int n) {
    if (curr > n) {
        return;
    }
    System.out.println(curr);
    curr++;
    print1toN(curr, n);
}
```

SS → Top  
S  
H  
E

Recursion Tree

LI FO  
FI LO

Recursion Tree



```
public static void print1toN(int curr, int n) {
    if (curr > n) { //count>3
        return;
    }
    print1toN(curr + 1, n);
    System.out.println(curr);
}
```

```
public static void print1toN(int curr, int n) {
    if (curr > n) { //count>3
        return;
    }
    print1toN(curr + 1, n);
    System.out.println(curr);
}
```

```
public static void print1toN(int curr, int n) {
    if (curr > n) { //count>3
        return;
    }
    print1toN(curr + 1, n);
    System.out.println(curr);
}
```

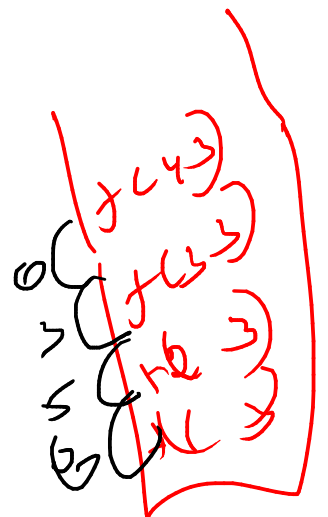
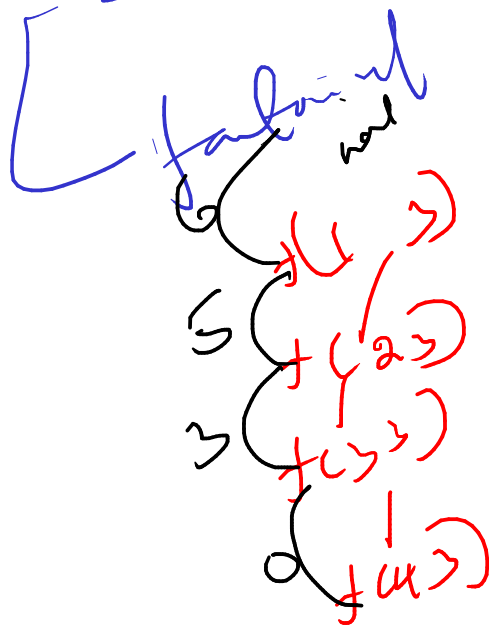
```
public static void print1toN(int curr, int n) {
    if (curr > n) { //count>3
        return;
    }
    print1toN(curr + 1, n);
    System.out.println(curr);
}
```

Recurs  $\rightarrow$  fn calling itself  
until some condition  
is met

Base condition -  
condition that stops  
recursion calls

Recursion Tree  
stack for calls

Sum of N nbs



```

public static int sumOfNumbers(int curr, int n){
    if(curr>n){
        return 0;
    }
    int sum = curr + sumOfNumbers(curr+1, n);
    return sum;
}

```

Handwritten annotations: Red numbers 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100. Red arrows pointing to the recursive call.

```

public static int sumOfNumbers(int curr, int n){
    if(curr>n){
        return 0;
    }
    int sum = curr + sumOfNumbers(curr+1, n);
    return sum;
}

```

Handwritten annotations: Red numbers 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100. Red arrows pointing to the recursive call.

```

public static int sumOfNumbers(int curr, int n){
    if(curr>n){
        return 0;
    }
    int sum = curr + sumOfNumbers(curr+1, n);
    return sum;
}

```

Handwritten annotations: Red numbers 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100. Red arrows pointing to the recursive call.

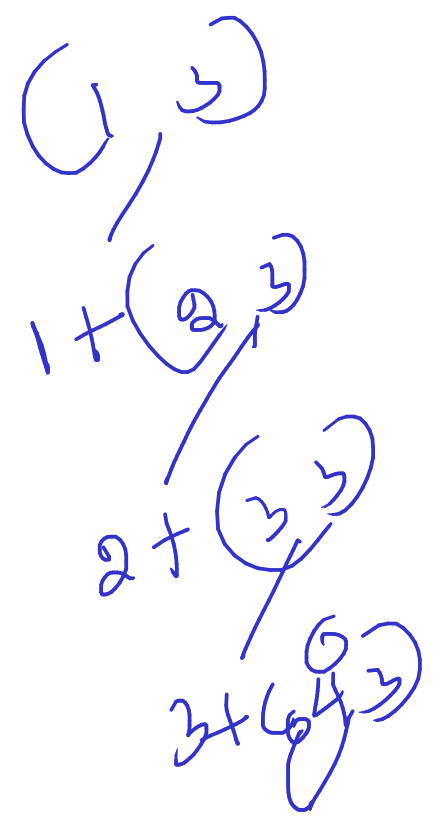
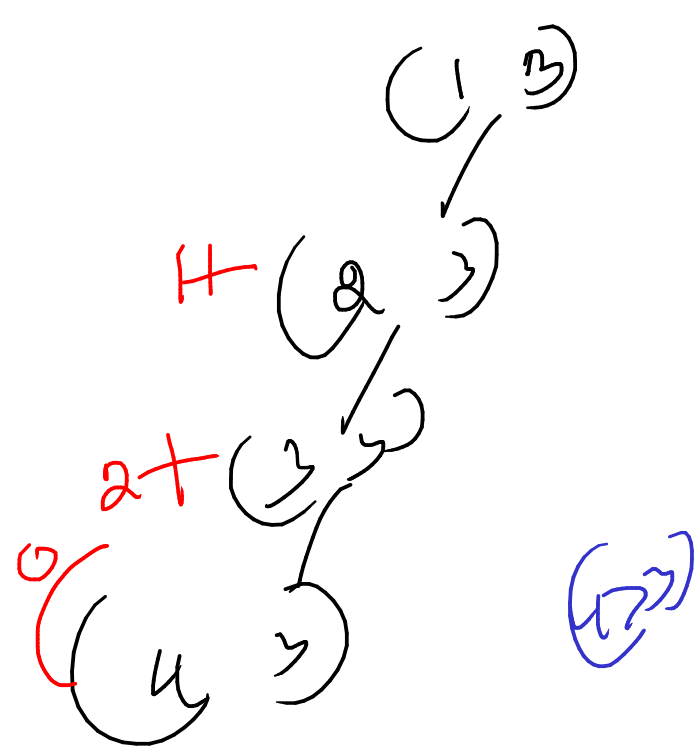
```

public static int sumOfNumbers(int curr, int n){
    if(curr>n){
        return 0;
    }
    int sum = curr + sumOfNumbers(curr+1, n);
    return sum;
}

```

Handwritten annotations: Red numbers 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100. Red arrows pointing to the recursive call.

10 + (20 + (50 + 60))



next

4 Friday

1) Datatype / variable / Hollomond

2) Loops / control states

3) functions

4) Array 1D array / 2D array

5) Recursion

[ 1) Intro  
2) Data? → variable  
3) Polymorphism

Need for Array  
Bank in 20 cars

h=  
h=10  
h=100

int a=10  
b=20  
c=30

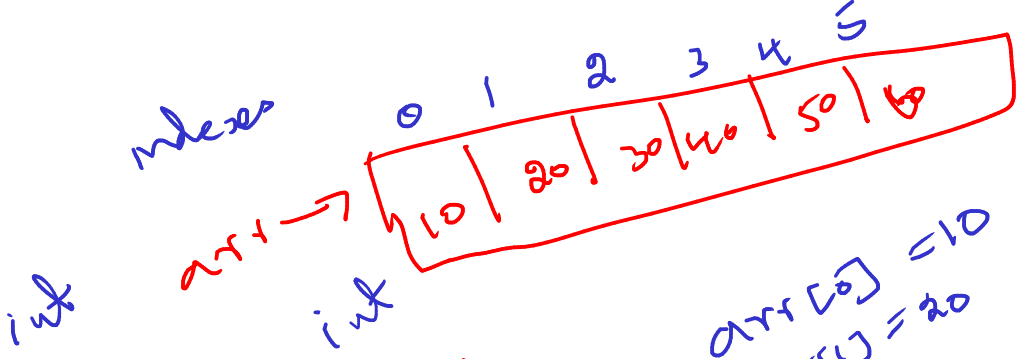
d=40  
e=50  
f=60  
g=70

Index

(a)=10

Array Homogeneous

datatype →  
similar type of  
data

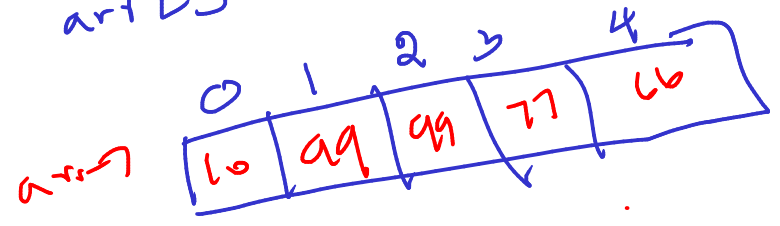


access → index

has → allocated set  
new memory

arr[0] = 10  
arr[1] = 20

int arr[] = new int [5] sum



→ last n g

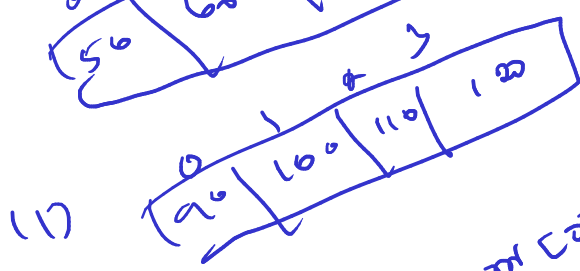
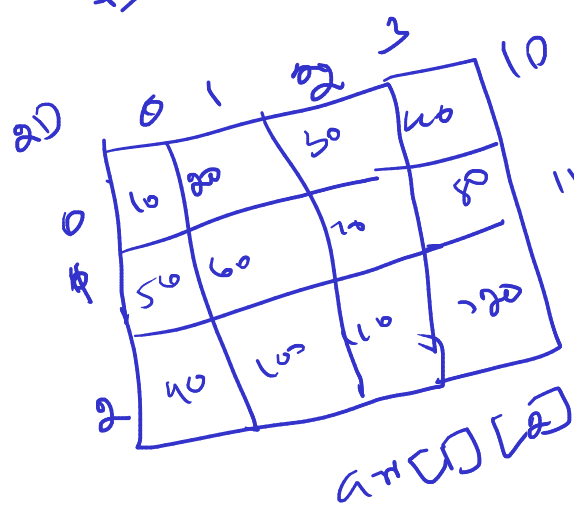
⑤  
0 → 5

can't modify size

$(0 \rightarrow n-1)$  10

2D Re

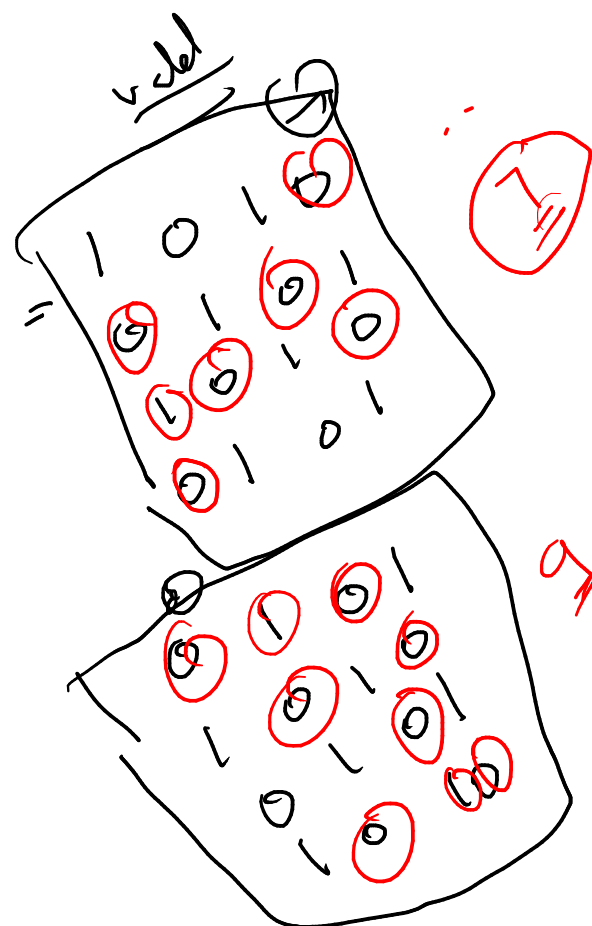
1D array



arr[0] ⇒ arr[0][0]

① ২০১৮

97716

 ~~$x + 4$~~ 

$(x) \cdot 1/2$

5/

A hand-drawn 4x4 grid containing binary values (0 and 1). The grid is tilted and has a rough, sketched appearance. The values are distributed as follows:

0	1	0	1
1	0	1	0
0	1	0	1
1	0	1	0

0 1 2 3  
0 1 2 3  
0 1 2 3 4  
1 2 3 4 5  
2 3 4 5 6  
3 4 5 6 7  
4 5 6 7 8  
5 6 7 8 9  
6 7 8 9 10  
7 8 9 10 11  
8 9 10 11 12  
9 10 11 12 13  
10 11 12 13 14  
11 12 13 14 15  
12 13 14 15 16  
13 14 15 16 17  
14 15 16 17 18  
15 16 17 18 19  
16 17 18 19 20  
17 18 19 20 21  
18 19 20 21 22  
19 20 21 22 23  
20 21 22 23 24  
21 22 23 24 25  
22 23 24 25 26  
23 24 25 26 27  
24 25 26 27 28  
25 26 27 28 29  
26 27 28 29 30  
27 28 29 30 31  
28 29 30 31 32  
29 30 31 32 33  
30 31 32 33 34  
31 32 33 34 35  
32 33 34 35 36  
33 34 35 36 37  
34 35 36 37 38  
35 36 37 38 39  
36 37 38 39 40  
37 38 39 40 41  
38 39 40 41 42  
39 40 41 42 43  
40 41 42 43 44  
41 42 43 44 45  
42 43 44 45 46  
43 44 45 46 47  
44 45 46 47 48  
45 46 47 48 49  
46 47 48 49 50  
47 48 49 50 51  
48 49 50 51 52  
49 50 51 52 53  
50 51 52 53 54  
51 52 53 54 55  
52 53 54 55 56  
53 54 55 56 57  
54 55 56 57 58  
55 56 57 58 59  
56 57 58 59 60  
57 58 59 60 61  
58 59 60 61 62  
59 60 61 62 63  
60 61 62 63 64  
61 62 63 64 65  
62 63 64 65 66  
63 64 65 66 67  
64 65 66 67 68  
65 66 67 68 69  
66 67 68 69 70  
67 68 69 70 71  
68 69 70 71 72  
69 70 71 72 73  
70 71 72 73 74  
71 72 73 74 75  
72 73 74 75 76  
73 74 75 76 77  
74 75 76 77 78  
75 76 77 78 79  
76 77 78 79 80  
77 78 79 80 81  
78 79 80 81 82  
79 80 81 82 83  
80 81 82 83 84  
81 82 83 84 85  
82 83 84 85 86  
83 84 85 86 87  
84 85 86 87 88  
85 86 87 88 89  
86 87 88 89 90  
87 88 89 90 91  
88 89 90 91 92  
89 90 91 92 93  
90 91 92 93 94  
91 92 93 94 95  
92 93 94 95 96  
93 94 95 96 97  
94 95 96 97 98  
95 96 97 98 99  
96 97 98 99 100

1) halo ↪

15 → 1 3 5 15

16 → 1 2 4 8 16

16/4  
= 0

1 → n

oob

in (n/2) ~ 16/4

sorten

36

36  
18 24/2  
12 24/3  
9 24/4  
6

1 → 36

36 → 6  
√36

77

2 → 57

5-37

✓ 1, itself

1, 20

1 → n

① → ⑦

2 → n-1

2 → n-1

2 → 16

ref. in (17/2) = 20 X

$P_{\text{rim}} \ni O(f) \rightarrow O(\text{Swit})$

Prime sieve  $\rightarrow O(n \log(\log n))$

Prime

0	1	2	3	4	5	6	7
1	2	3				12	36

$P_1$

$36/1 = 36$   
 $36/2 = 18$

$P_2$

$5$

⑤  
②  
③

$$\begin{aligned} 36/1 &= \\ 36/2 &= 17 \\ 36/3 &= 12 \end{aligned}$$



16  
factors  $\Rightarrow$  1 2 4 8 16

Per me  $\Rightarrow a$

$\underline{\underline{21}}$      $2 \rightarrow 3$      $7$      $\underline{\underline{27}}$      $3$      $3$      $3$   
 $5 \rightarrow \underline{\underline{5}}$

$17 \rightarrow 17$

~~✓ 71~~

[illegible]

TC

math

1) factors  $\rightarrow$  prime factors  
2)

TC

math

Brute / Better / Optimal

1) factorization  
 $\rightarrow$  prime factors

2) primes  $\rightarrow$   
3) gcd  
4) gcd

1) factors, prime factors

2) primes

3) gcd

4) gcd

$10 \rightarrow 1 \quad \underline{2} \quad 5 \quad 10$   
 $16 \rightarrow 1 \quad 2 \quad 4 \quad 8 \quad 16$   
 $17 \rightarrow 1 \quad 17$

$10 \cdot 2 =$   
 raily  
 0

(1) 2 3 4 5 6 7 8 9 (10)

$10 \cdot 1 =$

$10 \cdot 6 = 4$

5

$1 \rightarrow 36/1$   
 $2 \rightarrow 36/2$

$\underline{36}$   
 1  
 2  
 3  
 4  
 6

$36$   
 $18$   
 $12$   
 $9$   
 $36/1$   
 $36/2$   
 $36/3$   
 $36/4$   
 $36/6$

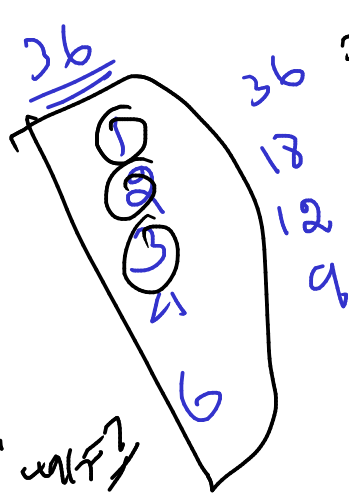
$1 \rightarrow$

$n/a$

$5n$

$36 \rightarrow 6$   
 $36 \rightarrow 56$

$u_1$   
 $i \rightarrow 7$



$1 \rightarrow 36$

$n = 10$

divs

$(0 \rightarrow)$

1 2 5 10

$10 : 2 = 5$

$10 : 3 = 3.33$   
 $1 =$

$(\tau \text{ or } \tau = 0)$

$16 \rightarrow$  1 2 4 8 16

$\frac{8}{1}$   $\frac{0}{2}$   $\frac{2}{3}$   $\frac{0}{4}$   $\frac{3}{5}$   $\frac{2}{6}$   $\frac{1}{7}$   $\frac{0}{8}$

$O(n) > \Theta(\log n)$

$u_1 > 7$

$100 > 10$

27

$36/2$

$n/3$

$$\frac{5n}{275n}$$

$$\frac{5n}{36/2}$$

$$\frac{36}{1 \rightarrow 18}$$

$$\frac{36}{2 \rightarrow 18}$$

$$\frac{36}{3 \rightarrow 12}$$

$$\frac{36}{4 \rightarrow 9}$$

$$\frac{36}{6 \rightarrow 6}$$

$$\frac{36}{1 \rightarrow 36}$$

$$\frac{36}{2 \rightarrow 18}$$

$$\frac{36}{3 \rightarrow 12}$$

$$\frac{36}{4 \rightarrow 9}$$

$$\frac{36}{6 \rightarrow 6}$$

$$\frac{36}{1 \rightarrow 36}$$

$$\frac{36}{2 \rightarrow 18}$$

$$\frac{36}{3 \rightarrow 12}$$

$$\frac{36}{4 \rightarrow 9}$$

$$\frac{36}{6 \rightarrow 6}$$

Prime

X

= 0 2

$$4 \times 2$$

$$16 \times 2 \times 2 \times 2 \times 2$$

$$9 \times 3$$

Indecent of one

prime  $\rightarrow$  2 divides  
1, it set

1, set

Prime (not ?)

$$2 \rightarrow n-1$$

1, 1

$(6 \rightarrow$   
 $13 \quad 14 \quad 15 \quad 16 \quad 17$   
 $\textcircled{2}$

$\textcircled{19} \quad 20 \quad 21 \quad 22 \quad 23$   
 $\swarrow \quad \searrow \quad \swarrow \quad \searrow$

Sie hat  $n-1$  fuhrt nach eine

$14 \rightarrow \textcircled{13}$

$\textcircled{23} \quad \textcircled{2} \quad 24 \quad 25$   
 $\swarrow \quad \searrow \quad \swarrow \quad \searrow$   
 $\textcircled{4}$   
 $2$

$11 \quad 12 \quad 13$   
 $\textcircled{11} \quad \textcircled{13}$

1) Prim. sieve  $\rightarrow O(\log \log n)$

2) Prime factor as number

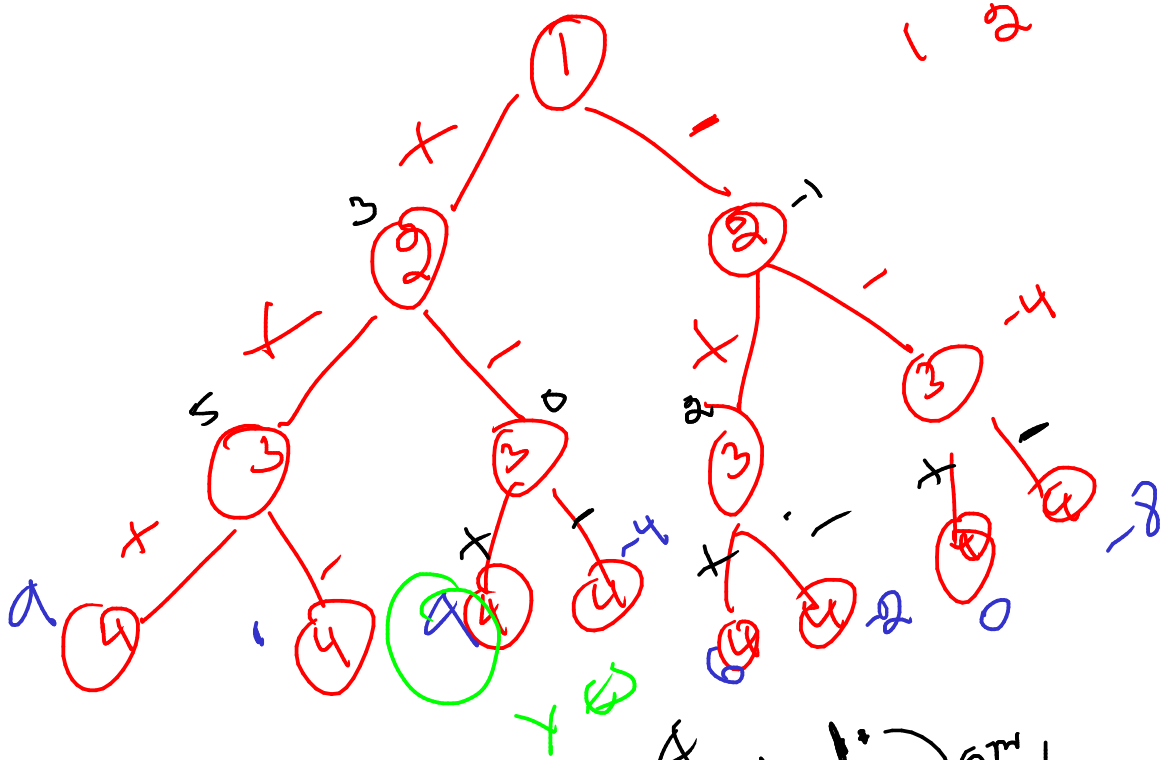
3) result 1

44  
1 2 3 4

$$1+2+3+4 = \textcircled{4} \quad n=4$$

$$( \quad \times \quad 2 \quad \times \quad 3 \quad \times \quad 4 )$$

~~4~~



f (sum, index, arr, ans)  
if (sum == ans) {  
 solve (i+1, ans)

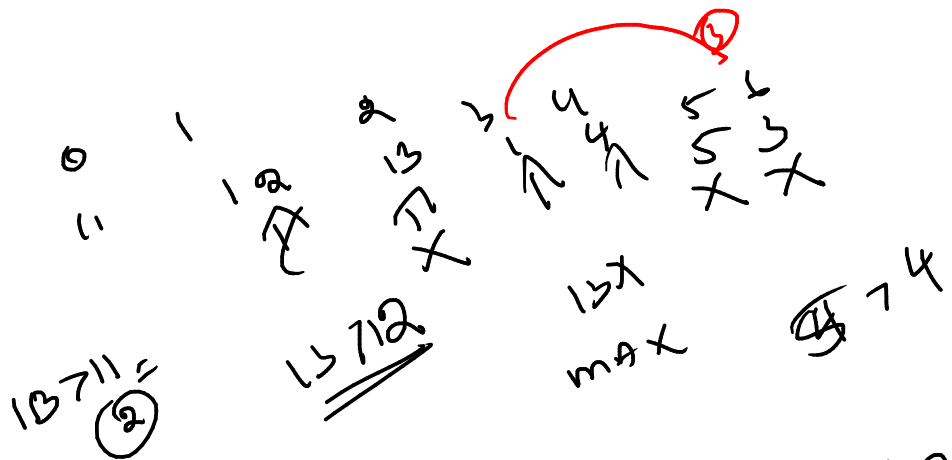
if (i > arr.length) { ans = f (sum + arr[i], i+1, ans)  
 ans = f (sum - arr[i], i+1, ans)

return ans



5 4 2 1  
 1 2 3 4 5

arr [0] 7 arr [1] 7



ans = -1

$i = 0, j = 0$

$(i < n \ \& \ j < n) \{$

$\text{if } (arr[i] < arr[j]) \{$

$cd = j - i$

update diff

$\text{else } \{$   
 $i++$   
 $j++$

$\text{return } (max)$

17 11 12 13 14 1 2 18 9

Highest  
common  
factor

GCD | HCF  
Greatest common divisor

LCM

12    1   2   3   4   6   12  
33    1   3   11   33

common  
1 (3)

HCF / LCM

25 → 1 5 25

largest di

GCD / HCF

1 50 → 1 2 3 5 6 10 15 25 30  
50 25 150

1)

common

common max

25  
150

min  
~~max~~ (25, 150)

25

25, 150

12    min (12, 33) = 12  
33

12/12 = 0  
33/12 = 9  
12/1 = 1

12, 33

12/3 = 0  
33/3 = 1

12  
4  
6  
9  
(3)

Euclid Algo  $\Rightarrow$   $\begin{matrix} B \div A \\ A \div B \end{matrix}$   $A \div B$

gcd(a,b)  
 $\text{if } (b == 0) \text{ ret } a;$   
 $\text{return gcd}(b, a \% b);$

$B \Rightarrow A$   
 $A \div B \Rightarrow$

when  $B \div 0$   $\rightarrow$  (A)

a	A	B	Rem
2	33	12	9
1	12	9	3
3	9	3	0

$12 \sqrt{3}$

A  
105

B  
252

$A \div B$   
 $105 \div 252$   
 105

1

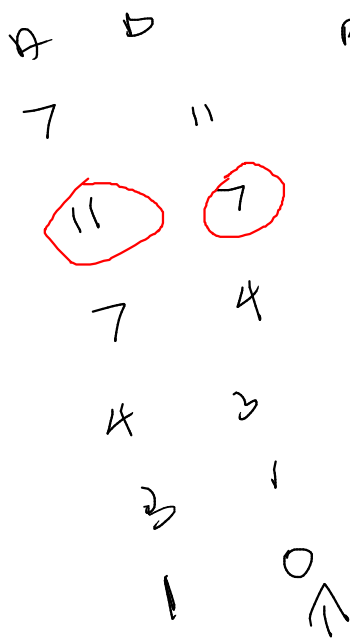
#(B)	A	B	A/B
2	252	105	42
2	105	42	21
2	42	21	0

$25 \div 2 \dots 105$

252

105

$\text{last min}(a,b)$



A/B

$7 \cdot 11$

$11 \cdot 7 = 4$

$7 \cdot 4 = 3$

$11 \cdot 3 = 1$

$3 \cdot 1 = 0$

LCM ?

$gcd(a, b) \neq lcm(a, b)$

$\Downarrow$

$(a \neq b)$

0

$gcd(a, b) \neq lcm(a, b) = a \neq b$

$lcm(a, b) = \frac{(a \neq b)}{gcd(a, b)}$

# OOPS - java

1) Basics of oops

2) collections

i) Vector

ii) ArrayList

iii) HashMap

HashSet

Sorting

searching

## OOPS

object oriented program

1) functional program → C

2) oops → C++, java

## student

1) Roll NO

2) Parents

3) Name

object? ⇒ bind with data  
connect

Real world entities

(logical) class → car

object/instance

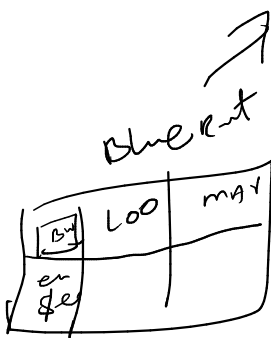
Blueprint / sketch

Actual obj / exists in real world


physical

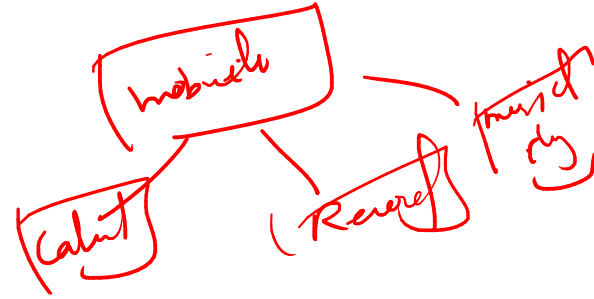
Bmw

Here



## Pillars goes

- 1) Abstraction → provide essential
- 2) Encapsulation →  → class
- 3) Inheritance → acquiring properties of another class
- 4) Polymorphism  
↳ one object → many forms

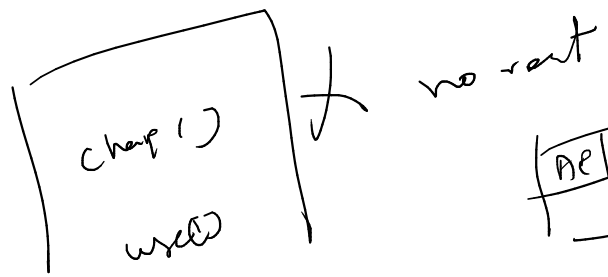


## Constructor Rules

- 1) No return type (void not allowed)
- 2) name → same as class name
- 3) gets called automatically when objects are created
- 4) constructor → can be overloaded
- 5) used → to initialize variables







AP	CS	KE
----	----	----

1) class → Blueprint → catalogue

2) object → real world entity

Actual  
size

- 1) Wrapper collection .. class
- 2) Vector
- 3) ArrayList
- 4) Pair

Primitive datatype

int a = 10;

double c = 20.5;

char gender = 'm';

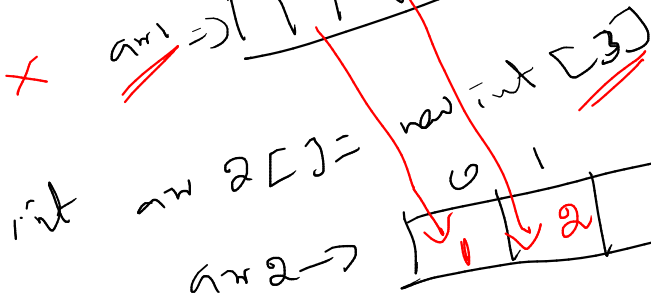
not an object

8 → Primitive datatypes

int Integer  
float Float  
char  
double  
byte  
Boolean  
long  
short  
String

array

int arr1[3] = new int[3];

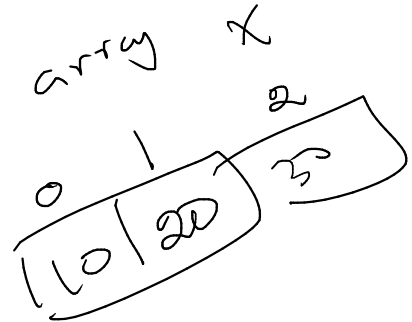


3

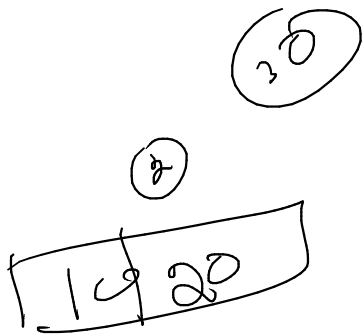
arr1 = arr2

Arr  $\rightarrow$  list / value  
 $\downarrow$

dynamic

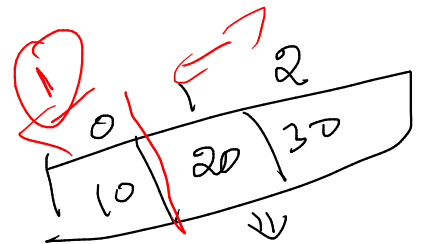


arr  $\Rightarrow$   
 $\uparrow$



Analysis  
 $\downarrow$

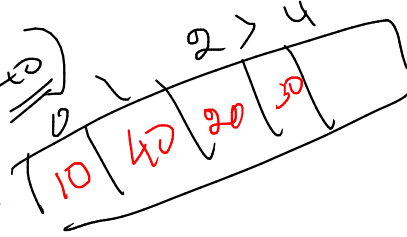
1) add  $\rightarrow$  o.c.i)  $\rightarrow$  a.w.c.i) (3)  $\Rightarrow$  6



insert int  
between

2) set  $\Rightarrow$   
 $\downarrow$   
 insert in  
 between

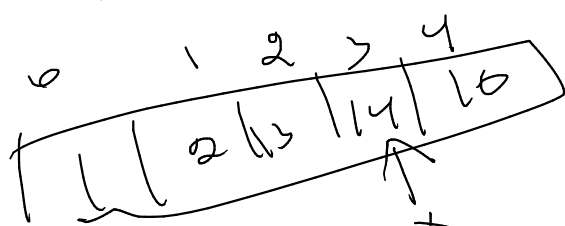
list.set ( o.c.i) u.c.i)



3) arr[0]  $\Rightarrow$  ?

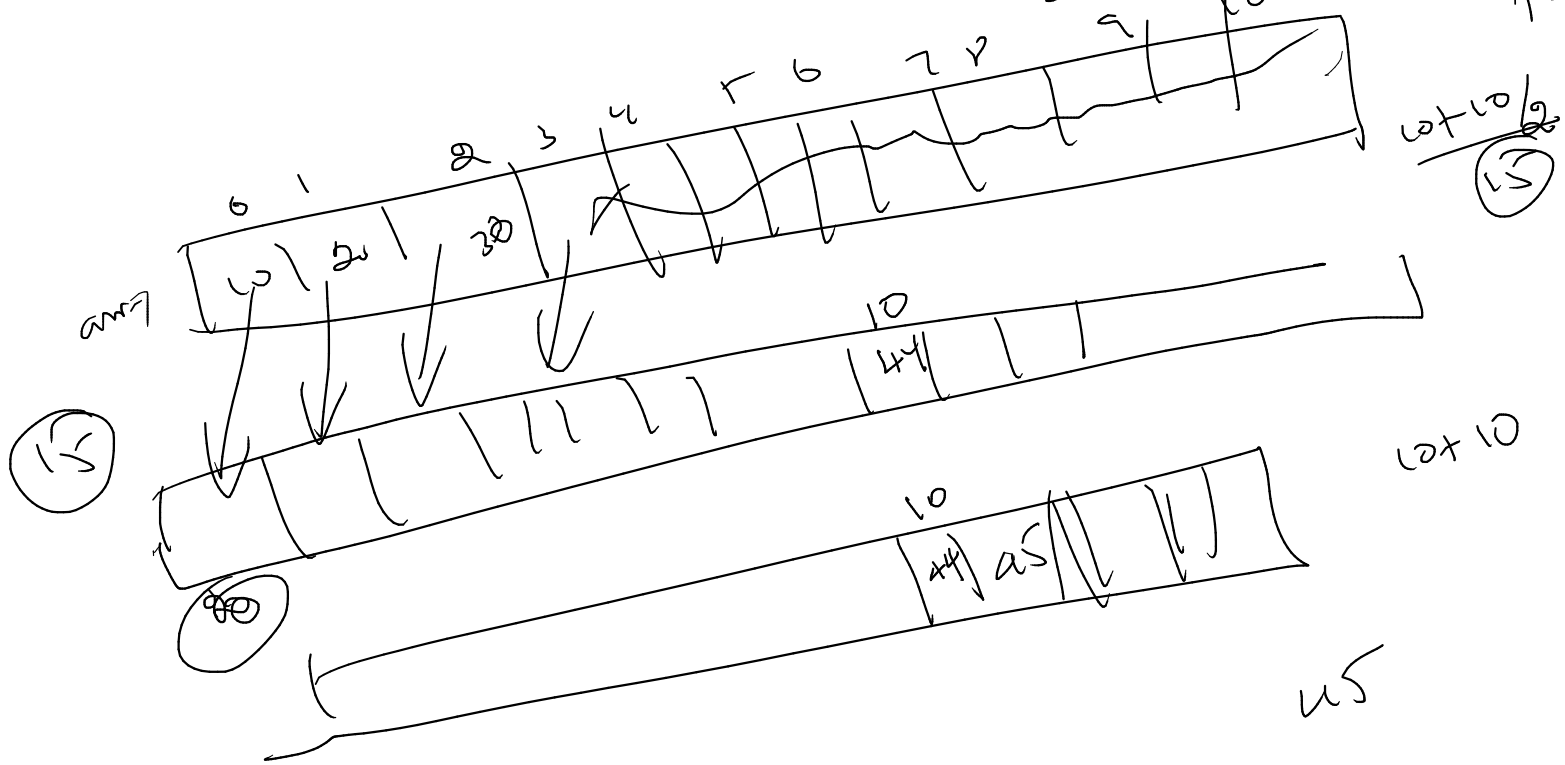
list.set(0)  $\Rightarrow$  o.c.i)

arr  $\rightarrow$



for loop?

working 50% Array vector ~~(100%)~~ size = 10 li: add (the)



you? s2 +1

$$\text{Arry} = s + s/2$$

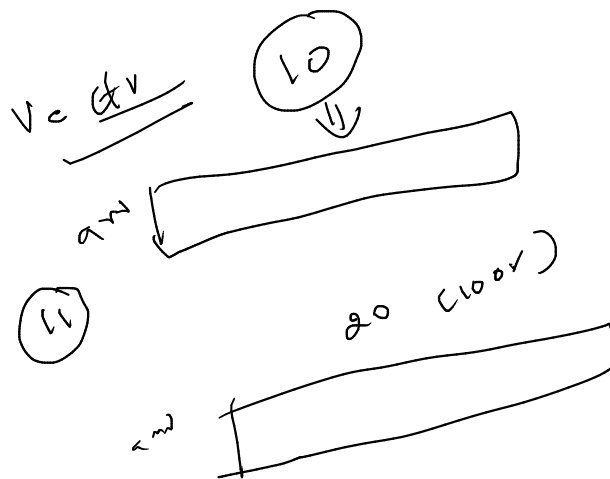
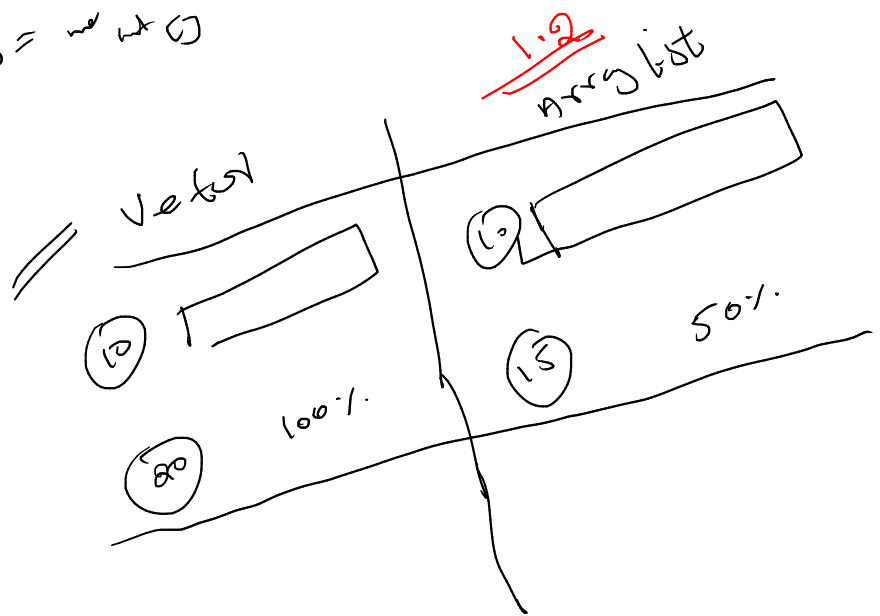
$$v = s + s$$

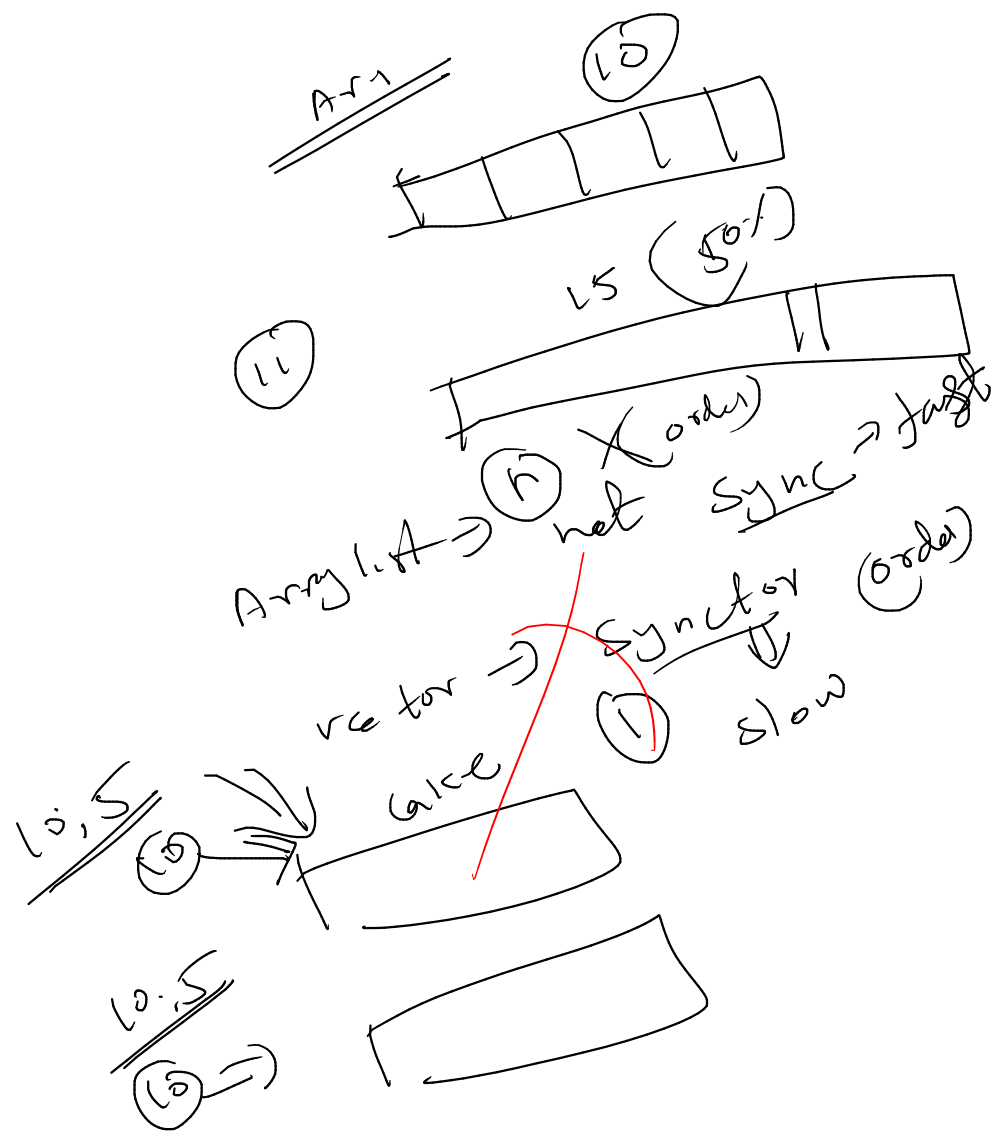
collecting  
Array List  
Vector  
 quiz

Set  
 Pair  
 Hashmap

collecting  
only Package / Library  
 A registry  
 out of the box

$arr[i] = \text{new int}()$





set

distinct

set

unique

union  
intersection

$\cup \Rightarrow$  merge

$\cap \Rightarrow$  common

$A = \{10, 20, 30\}$

$B = \{20, 40, 60\}$

$A \cup B = \{10, 20, 30, 40, 60\}$  merge

$A \cap B = \{20\}$  common

Set order  
1. HashSet -- Hashing - not predict  
2. TreeSet -- Tree - sorted order  
3. LinkedHashSet - LinkedList - inserted order

Input:

N = 5

10 30 20 10 20

Output:

10 20 30

print all unique numbers in sorted order

