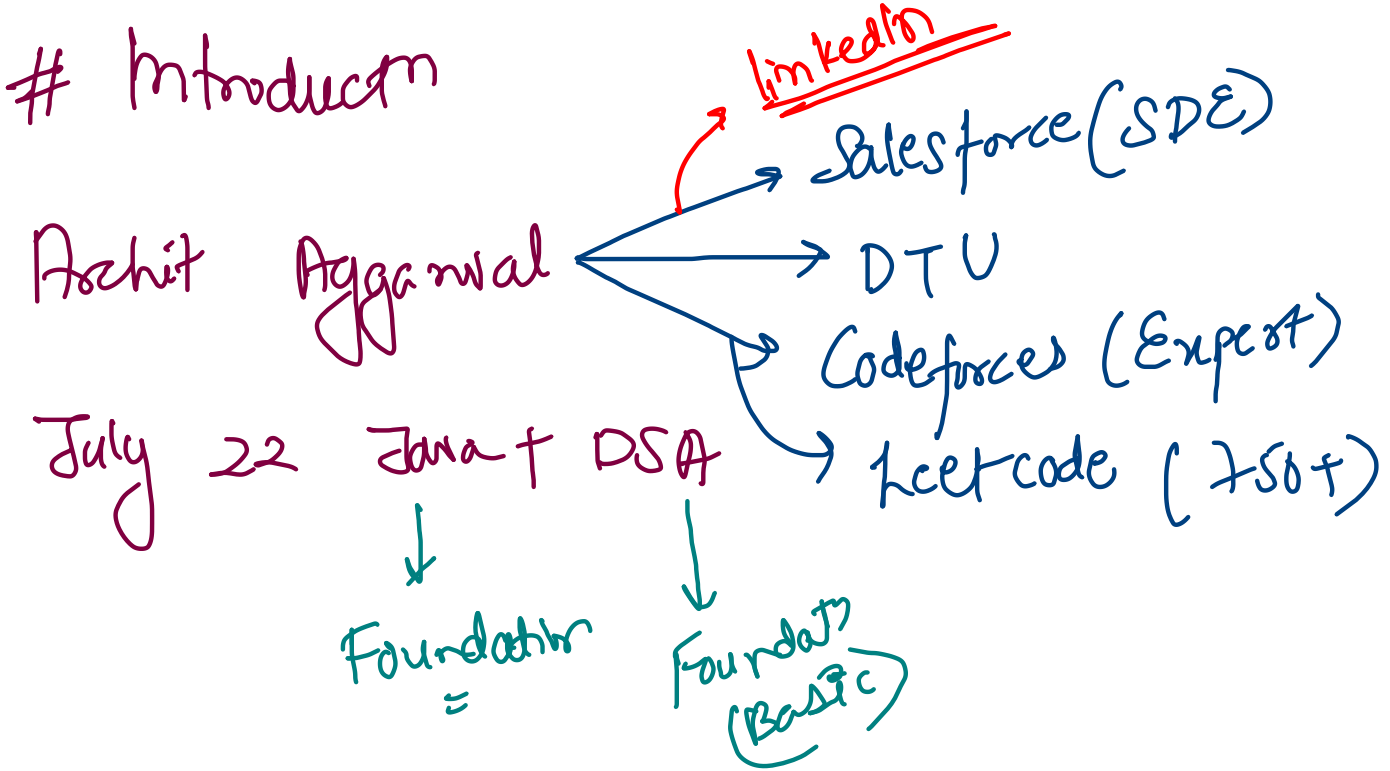
Introduction



5-15 LPA

Syllabus (3 months)

→ Java Programming language → If-else, for loops, Pattern

↳ Abdul Bari
↳ Java → Udemy (300 videos)

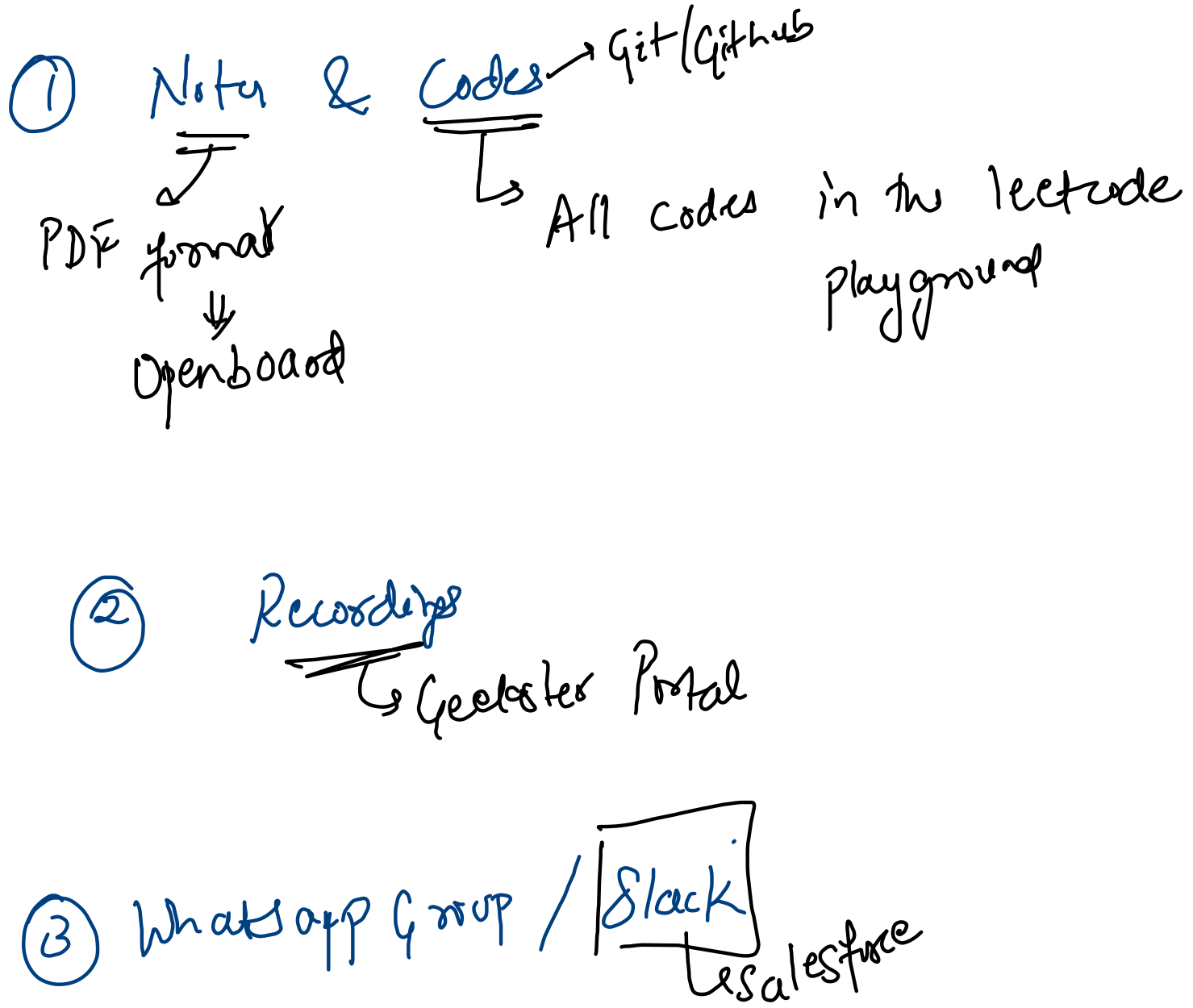
→ Arrays (1D & 2D), Strings & StringBuilder → Time & Space Complexity

→ Searching & Sorting → Two Pointers → PrefixSum

→ Object oriented Programming in Java + Projects

→ Collection framework → Stack & Queue, LinkedList, Hashmap & Heap

→ Recursion & Backtracking

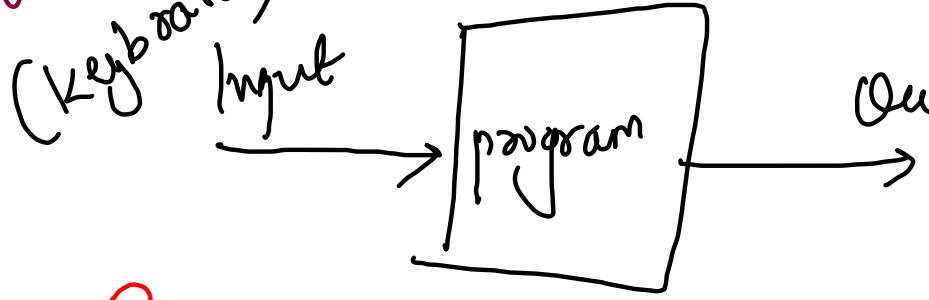

1 Notes & Codes
Git/Github
PDF format
Openboard
All codes in the leetcode playground
2 Recordings
Geekster Portal
3 Whatsapp Group / Slack
Salesforce

Integrated Development Environment (IDE)

Software

Offline
- VSCode
- Sublime Text
- Eclipse (Java)

Online
- leetcode
- GFG



→ Programmers
(code)
C ✓ ∝

1.5 - 2 ₹ per word
↑
Freelancing
→ Internicubit
→ GFG (Freedom city)
→ Repcoding
↳ Scaler
→ Unstop (D2C)
↳ Swiggy
→ Geeksforgeeks (year)

Java

High level Programming → English language

↓ Compilation (Compiler)

low level Programming
(Machine language) {Os and ls Binary}

↓

Memory/Heap (RAM)

Execution / Running Code
JDK (Java Development Kit)
JVM (Java Virtual machine)

10 + 15
Storage

Output

① System.out.println("Archit Aggarwal");

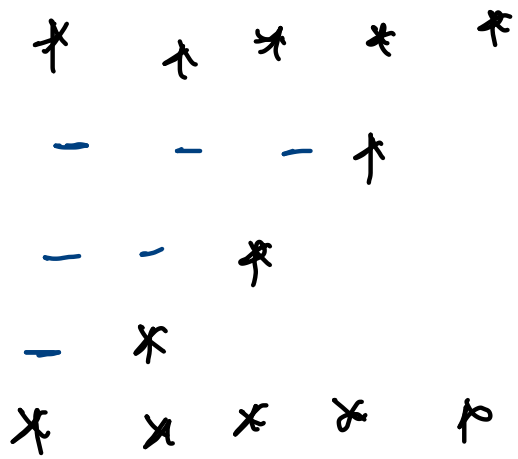
Capital

↳ Newline

② System.out.print("Geekster");

↳ Same line

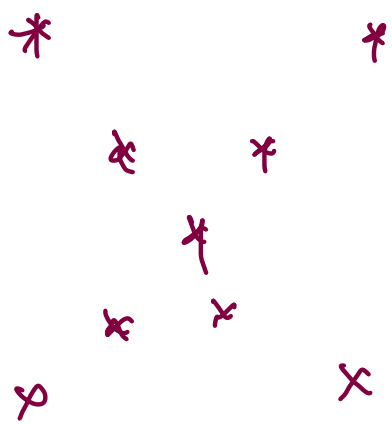
Pola 2



```
System.out.println("*****");
" println(" * ");
" println( );
" println( );
" println( );
```

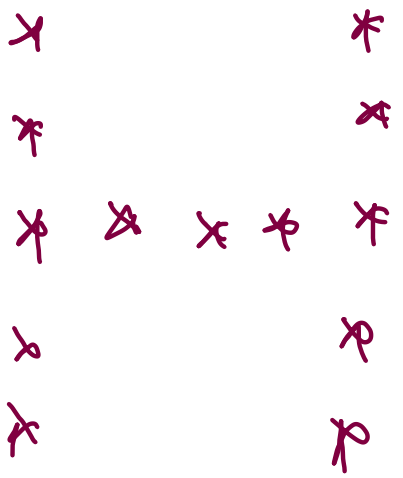
HW 1

Pola X



HW 2

Pola H



Live Teaching
20%-30%
Chats
Unmute (Raise Hands)

Teaching Assistant

↳ Shreya

- Doubts (7:30 - 8:00 PM) (1:1)
- Live Doubts (Chat)
- Hackerrank Assignments & Tests

Operators in Java

1. Arithmetic operators (+, -, /, *, %)
   - + : Addition / String concatenation
2. Comparison operators (<, >, <=, >=, ==, !=)
3. Assignment operator (=)
4. Logical operators (AND (&&), OR (||), NOT (!))

Logical Operators

for the && result to be true, both operands should be true

Truth table

AND (&&)

	T	F
T	T && T = T	T && F = F
F	F && T = F	F && F = F

Truth table

OR (||)

If any operand is true, then the result is true.

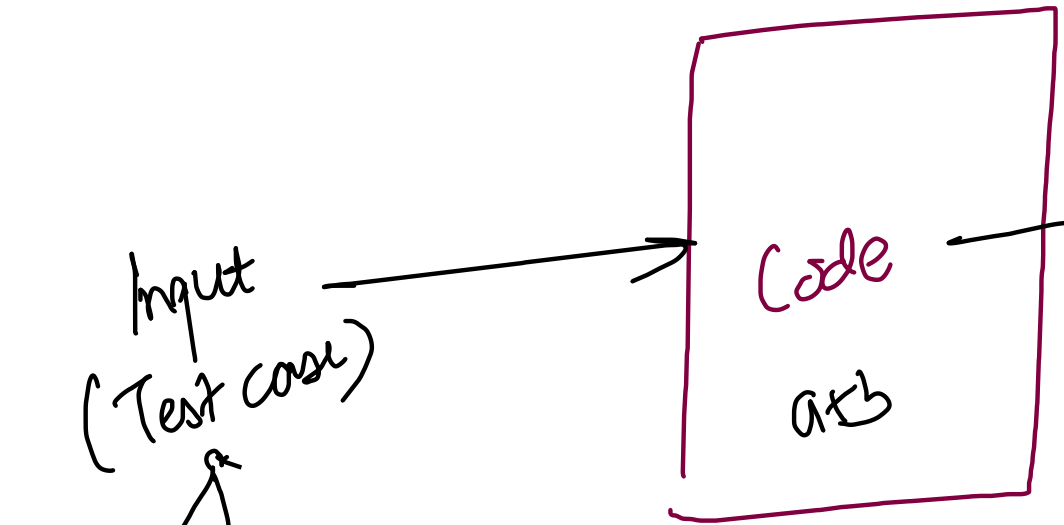
	T	F
T	T || T = T	T || F = T
F	F || T = T	F || F = F

Truth Table

NOT (!)

T	F
F	T

Hackerrank (Online Judge)



Sample
2 3

Hidden
-1 1
0 100000
-9999 9999
Corner
Edge
Boundary

Verdicts

1. Actual output == Expected output
   ⇒ Accepted
2. Compilation Error, Run time Error
3. Time limit Exceeded
4. Wrong answer
5. memory limit Exceeded

Question

→ Problem Statement

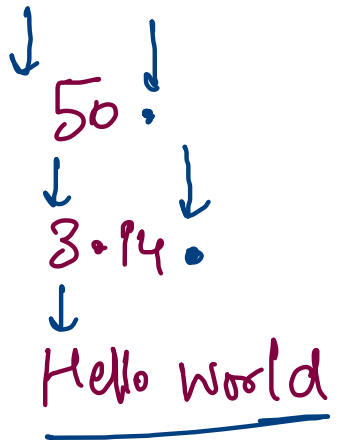
→ Input format

→ Output format

→ Constraints **

→ Time Limit Exceeded ☹

→ Accepted




```
Scanner scn = new Scanner(System.in);
int a = scn.nextInt();
double b = scn.nextDouble();
scn.nextLine();
String c = scn.nextLine();

System.out.println("String: " + c);
System.out.println("Double: " + b);
System.out.println("Int: " + a);
```

Geekster Evaluation (Referrals)

1. Attendance (CW Questions + % presence in Zoom + feedback ~~(Quiz)~~)
2. Homework Question (Hackerrank) ⇒ Assignment
3. Weekly Tests (Hackerrank ⇒ Teaching Assistant) Sunday
4. Suggestions from Instructor / Teaching Assistant

Conditional statements

Integer Age

$\geq 18 \Rightarrow$ true "Adult"

$< 18 \Rightarrow$ false "Below Age"

Syso (age >= 18)

Age	Salary	
$50 > 40$	$50000 \geq 30000$	Rich && Adult
$50 > 40$	$20000 < 30000$	Adult
$30 \leq 40$	$50000 > 12000$	Rich && Young
$30 \leq 40$	$10000 < 12000$	Young

Age > 40 && salary > 30k

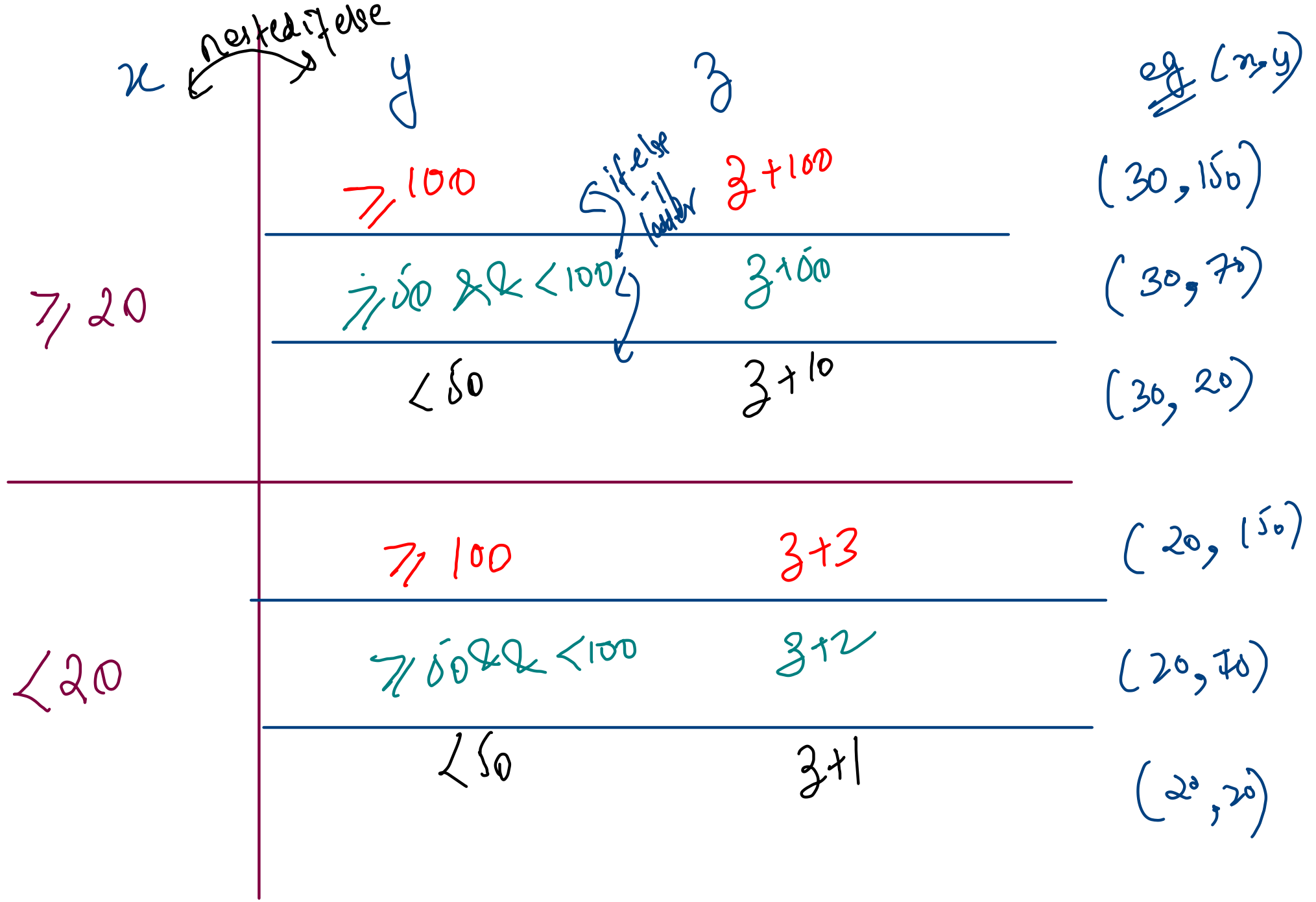
Age > 40 && salary $\leq$ 30k

Age < 40 && Salary > 30k

Age < 40 && Sal < 30k

Followup

Age	Salary	Gender	
$50 \geq 40$	$50000 \geq 30000$	male/female	Rich & Adult
$50 \geq 40$	$20000 < 30000$	male/female	Adult
$30 \leq 40$	$50000 \geq 12000$	m/f	Rich & Young
$30 \leq 40$	$10000 < 12000$	m/f	Young


x
nested if else
y
z
eg (x,y)
>= 100
if else - if ladder
z+100
(30,150)
>= 50 && < 100
z+50
(30,70)
>= 20
< 50
z+10
(30,20)
>= 100
z+3
(20,150)
>= 50 && < 100
z+2
(20,70)
< 20
< 50
z+1
(20,20)

characters

Symbols → ! @ # $ ₹ ... % ^ & * ...

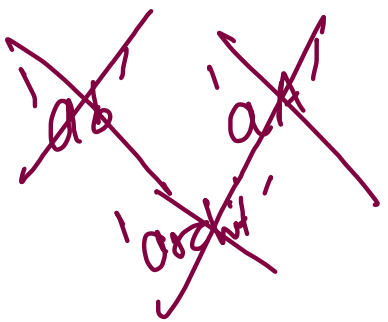
Special characters → tab, space, return, esc

emoji → 

English Alphabets

'a' - 'z'

'A' - 'z'



Devnagri (Hindi)

अ - ऊ

फ - ब - ह ....

digits

'0'

'1'

'2'

⋮

'9'

'00'

'99'

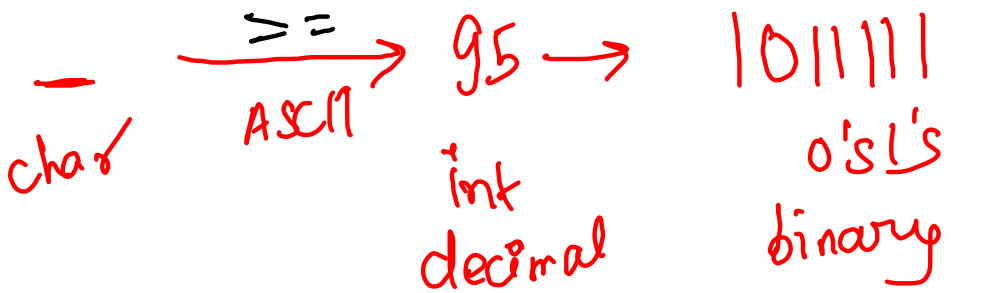
int marks = 95

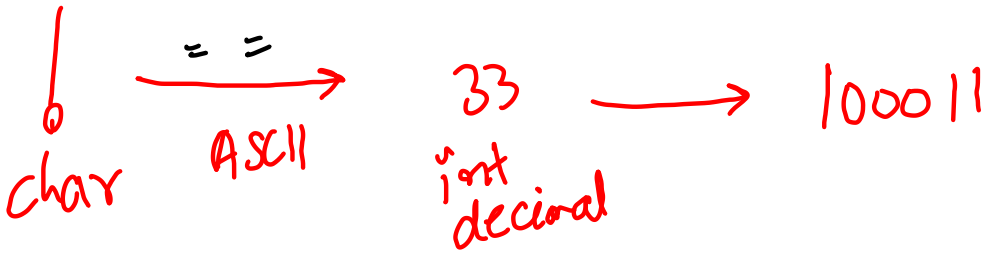
salary = 30000

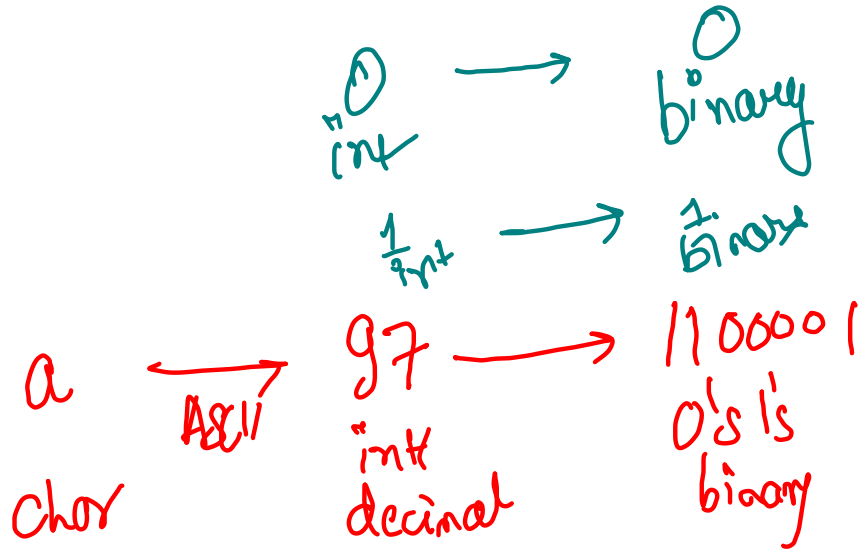
Unicode → All characters (ASCII + Hindi, Telugu, Emojis, ...)

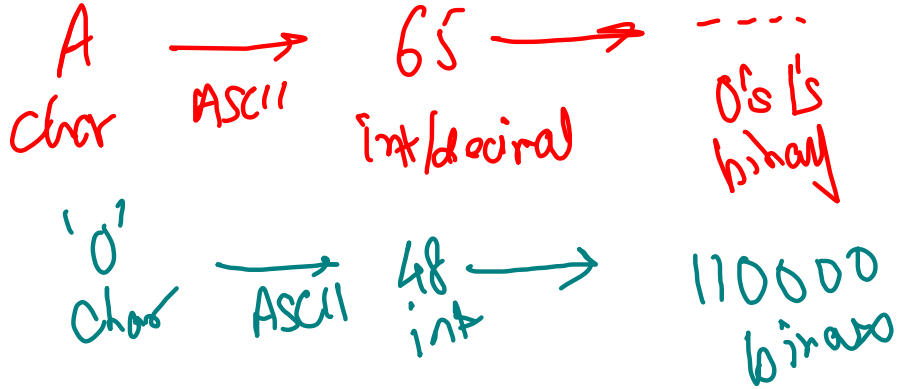
ASCII

American Standard Code for Information Interchange









Dec	Char	Dec	Char	Dec	Char	Dec	Char
0	NUL (null)	32	SPACE	64	@	96	`
1	SOH (start of heading)	33	!	65	A	97	a
2	STX (start of text)	34	"	66	B	98	b
3	ETX (end of text)	35	#	67	C	99	c
4	EOT (end of transmission)	36	$	68	D	100	d
5	ENQ (enquiry)	37	%	69	E	101	e
6	ACK (acknowledge)	38	&	70	F	102	f
7	BEL (bell)	39	'	71	G	103	g
8	BS (backspace)	40	(	72	H	104	h
9	TAB (horizontal tab)	41	)	73	I	105	i
10	LF (NL line feed, new line)	42	*	74	J	106	j
11	VT (vertical tab)	43	+	75	K	107	k
12	FF (NP form feed, new page)	44	,	76	L	108	l
13	CR (carriage return)	45	-	77	M	109	m
14	SO (shift out)	46	.	78	N	110	n
15	SI (shift in)	47	/	79	O	111	o
16	DLE (data link escape)	48	0	80	P	112	p
17	DC1 (device control 1)	49	1	81	Q	113	q
18	DC2 (device control 2)	50	2	82	R	114	r
19	DC3 (device control 3)	51	3	83	S	115	s
20	DC4 (device control 4)	52	4	84	T	116	t
21	NAK (negative acknowledge)	53	5	85	U	117	u
22	SYN (synchronous idle)	54	6	86	V	118	v
23	ETB (end of trans. block)	55	7	87	W	119	w
24	CAN (cancel)	56	8	88	X	120	x
25	EM (end of medium)	57	9	89	Y	121	y
26	SUB (substitute)	58	:	90	Z	122	z
27	ESC (escape)	59	;	91	[	123	{
28	FS (file separator)	60	<	92	\	124	|
29	GS (group separator)	61	=	93	]	125	}
30	RS (record separator)	62	>	94	^	126	~
31	US (unit separator)	63	?	95	_	127	DEL

+32
-32
+32
-32

ASCII table

'A' - 'Z'

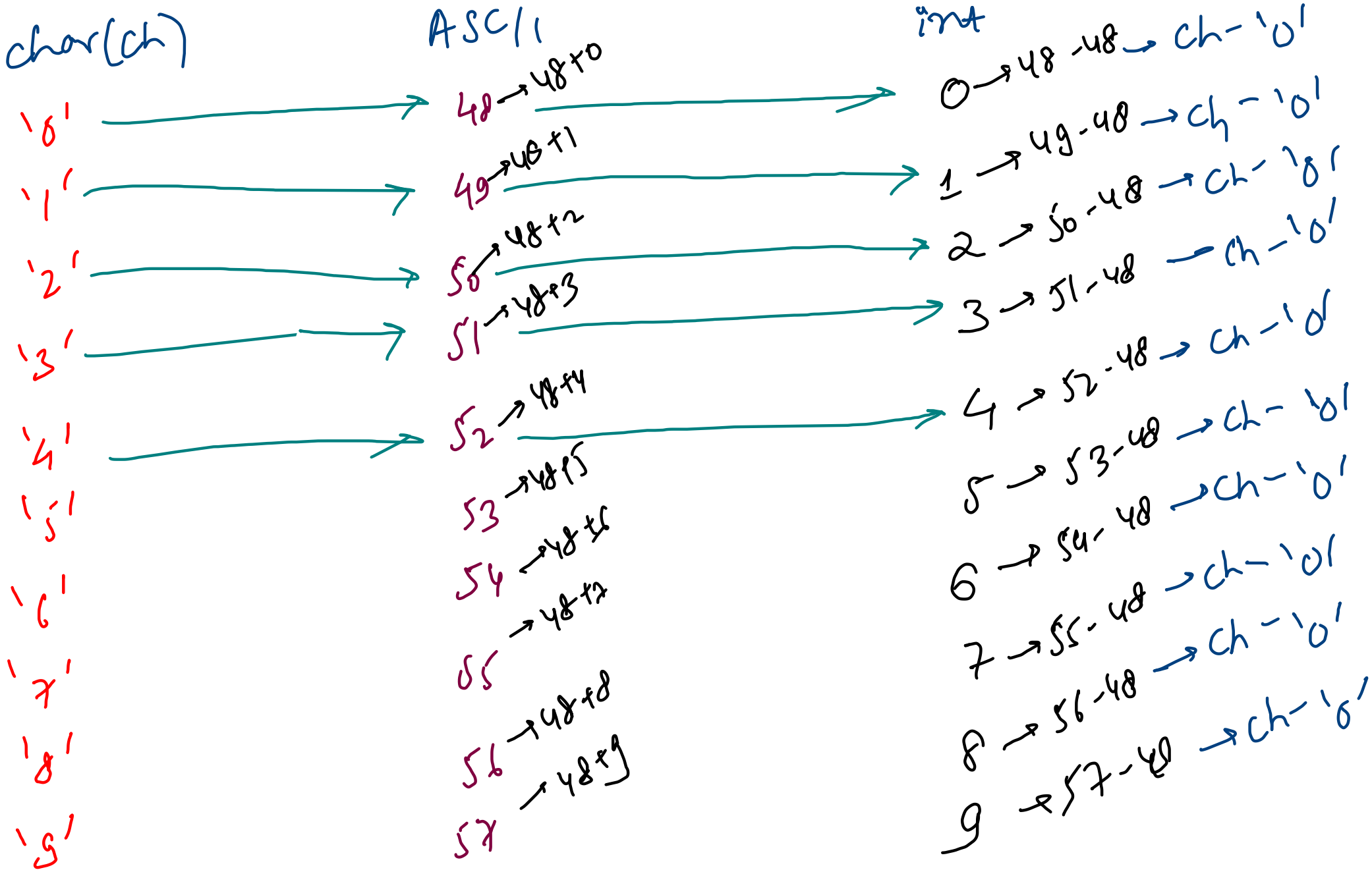
65 - 90

'a' - 'z'

97 - 122

'0' - '9'

48 - 57


char(ch)
ASCII
int
'0'
'1'
'2'
'3'
'4'
'5'
'6'
'7'
'8'
'9'
48 → 48+0
49 → 48+1
50 → 48+2
51 → 48+3
52 → 48+4
53 → 48+5
54 → 48+6
55 → 48+7
56 → 48+8
57 → 48+9
0 → 48 - 48 → ch - '0'
1 → 49 - 48 → ch - '0'
2 → 50 - 48 → ch - '0'
3 → 51 - 48 → ch - '0'
4 → 52 - 48 → ch - '0'
5 → 53 - 48 → ch - '0'
6 → 54 - 48 → ch - '0'
7 → 55 - 48 → ch - '0'
8 → 56 - 48 → ch - '0'
9 → 57 - 48 → ch - '0'

ch ⟶ ch+32

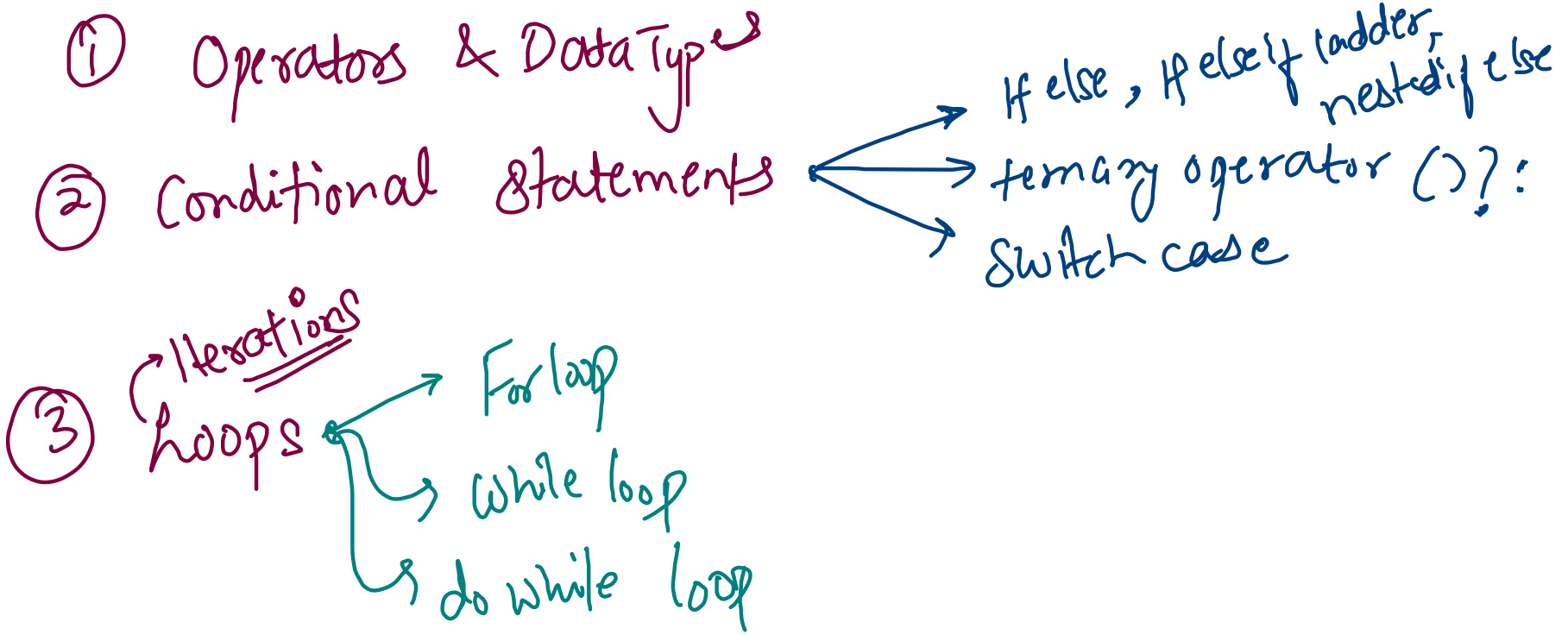
'A' ⟶ 'a'

Uppercase ⟷ lowercase

ch ⟶ ch-32

'a' ⟶ 'A'

lowercase → Uppercase


1 Operators & DataTypes
2 Conditional Statements
If else, If elseif ladder, nested if else
ternary operator ()?:
Switch case
3 Loops
Iterations
For loop
while loop
do while loop

Learn / Revise Pre-requisites {Mathematics}

- ★→ Arithmetic Progression, Geometric Progression (AP & GP)
- ★→ Permutations & Combinations, Factorial (${}^nC_r$, ${}^nP_r$)
- ★→ Number Theory (GCD/HCF, LCM, Prime No/Composite No)
- ★→ Logarithms
- → Geometry (2D matrix ← row / column)

For loops

idx = idx + 1

idx++

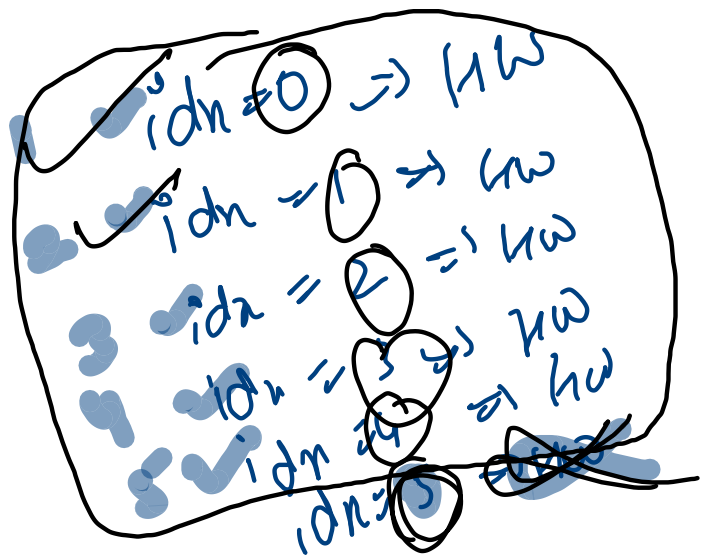
```
for ( initializatn ; terminatn ; updation ) {
```

initializatn → int idx = 0

terminatn → idx < 5

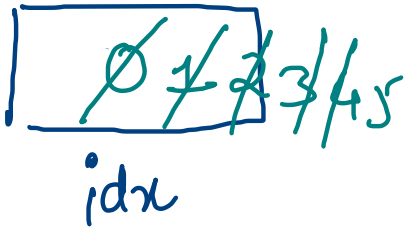
updation → idx++

```
    System.out.println("Hello world");
}
```



initialization termination updation

```
for(int idx=0 ; idx < 5 ; idx++){
    System.out.println("Hello World");
}
```



initialization ⇒ before entering for loop for the 1st time

termination ⇒ before entering the loop check if it is true enter the loop otherwise terminate (stop)

update ⇒ after executing all the statements, then update

idx < 5 : 0 < 5 : true : Hw 0

idx < 5 : 1 < 5 : true : Hw 1

idx < 5 : 2 < 5 : true : Hw 2

idx < 5 : 3 < 5 : true : Hw 3

idx < 5 : 4 < 5 : true : Hw 4

idx < 5 : 5 < 5 : false → stop

```
for ( int idx = 0 ; idx < 5 ; idx++ ) {
    System.out.println( idx );
}
```

idx ≤ 4

0
1
2
3
4

idx ≤ 4

N = 5

0
1
2
3
4

N = 3

0
1
2
3

N = 0

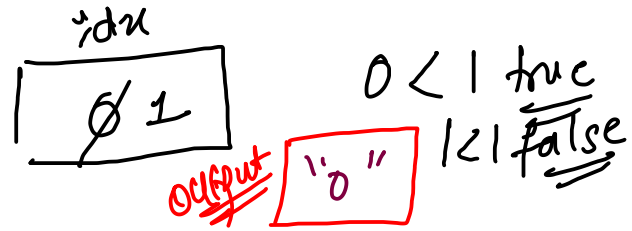
0

N = 10

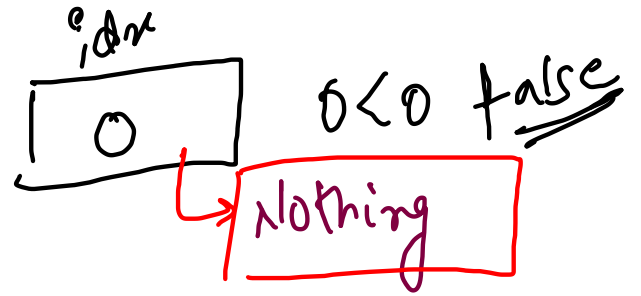
0
1
2
3
4
5
6
7
8
9
10

Question

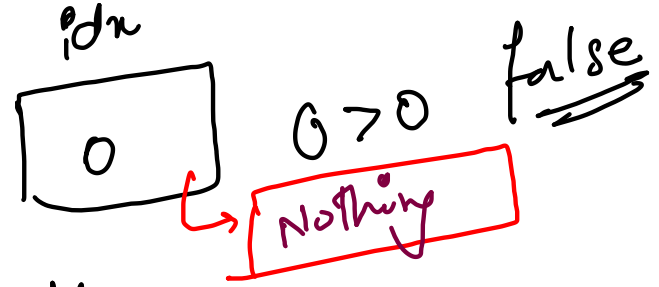
① `for ( int idx=0 ; idx < 1 ; idx ++)`



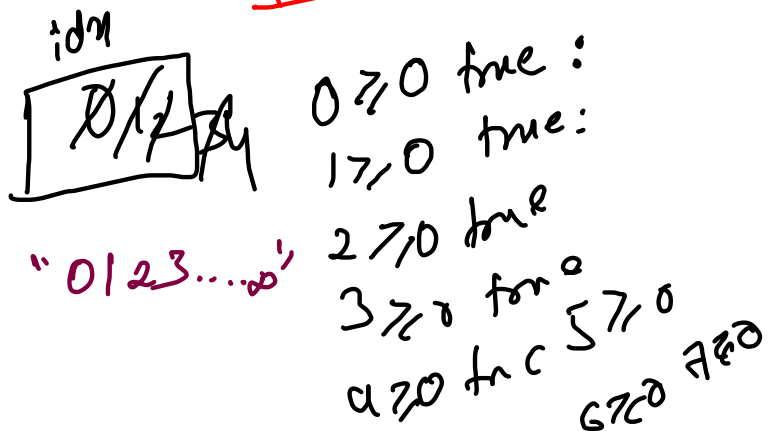
② `for ( int idx=0 ; idx < 0 ; idx ++)`

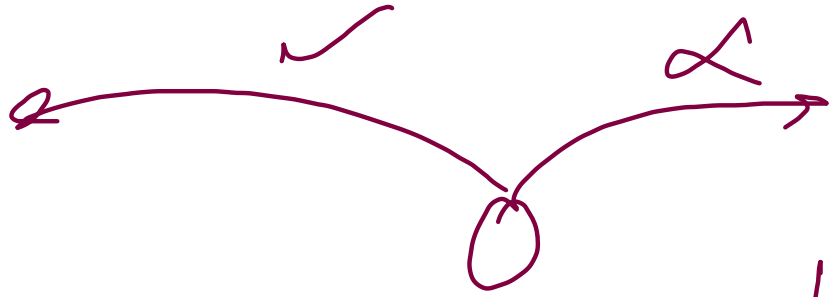


③ `for ( int idx=0 ; idx > 0 ; idx ++ )`



④ `for ( int idx=0 ; idx >= 0 ; idx ++ )`





Even

0, 2, 4, 6, 8, 10, ...

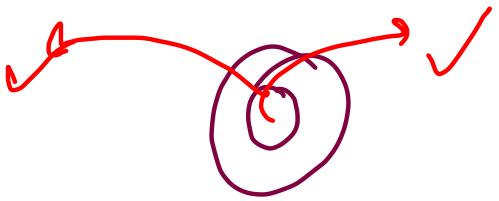
odd

1, 3, 5, 7, 9, 11, 13, ...

Prime

1, 2, 3, 5, 7, 11, 13, 17, ...

1 2 1 3 1 7

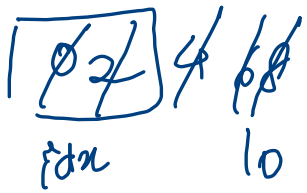


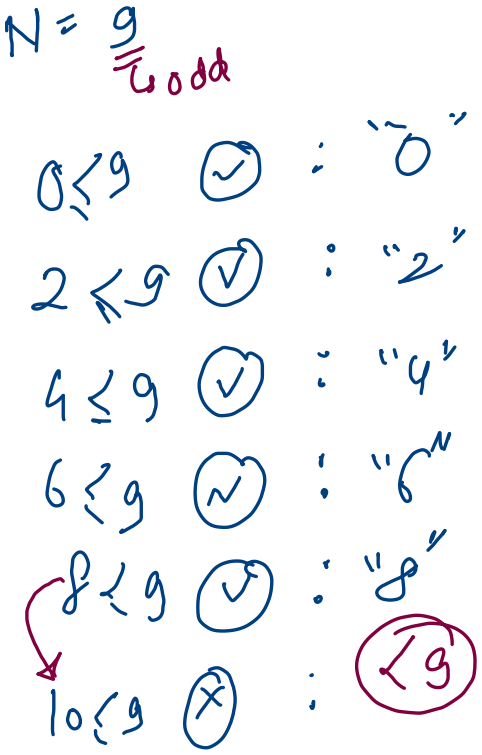
0 is neither nor Composite

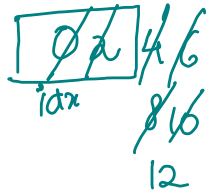
Composite

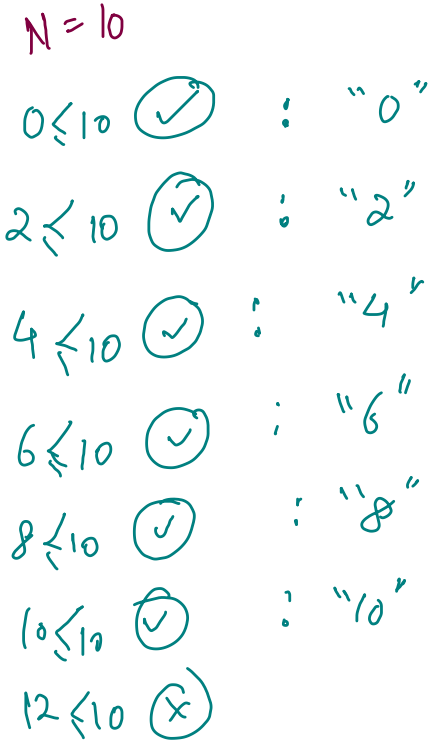
4, 6, 8, 9, 10, 12, ...

```java
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();

        for(int idx=0; idx<=n; idx=idx+2){
            System.out.println(idx);
        }
    }
}
```



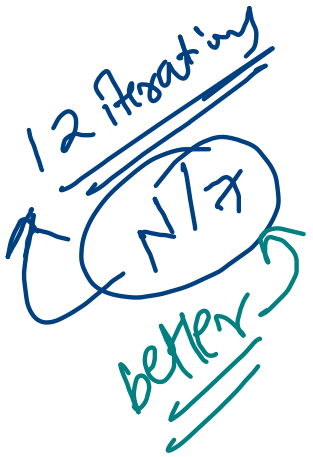






Code 1

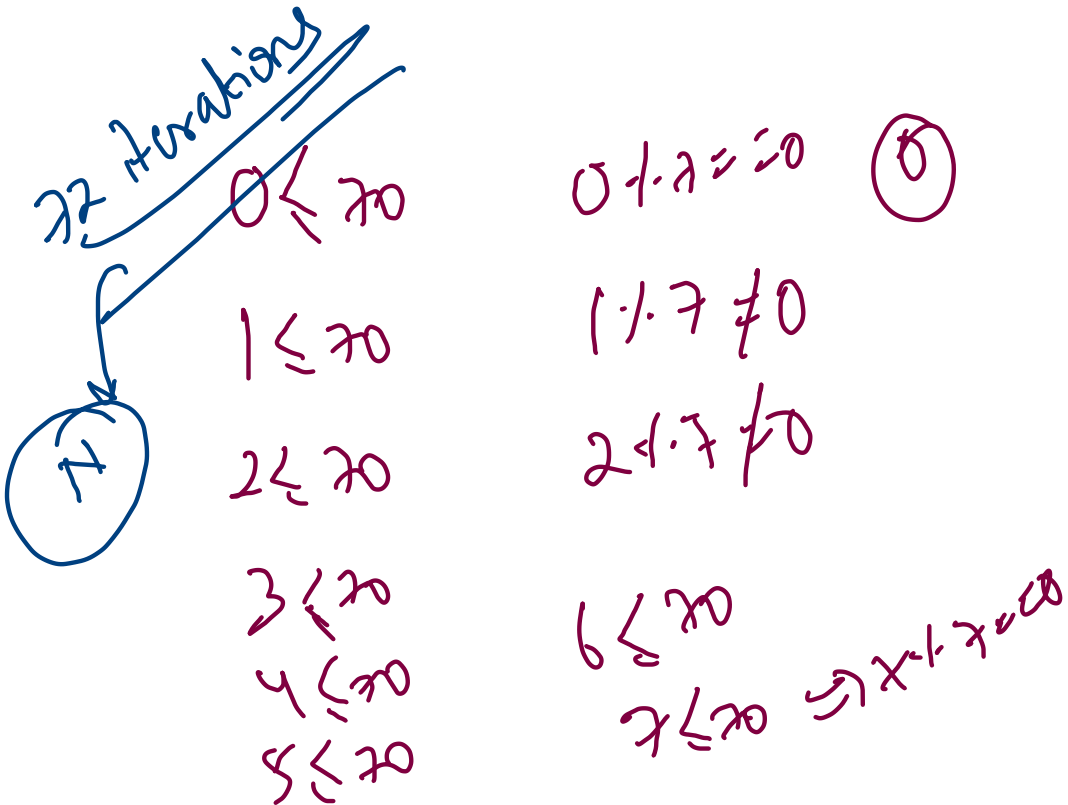
```
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();

    for(int idx=0; idx<=n; idx=idx+7){
        System.out.print(idx + " ");
    }
}
```



N = 70

$0 \le 70$ ⓪
$7 \le 70$ ⑦
$14 \le 70$ ⑭
$21 \le 70$ ㉑
$28 \le 70$ ㉘
㉟ ㊷ ㊾ ㊺ ㊸ ㉚

Code 2




```
public static void main(String[] args) {
    Scanner sc=new Scanner(System.in);
    int N=sc.nextInt();
    for(int i=0;i<=N;i++){
        if(i%7==0){
            System.out.print(i+" ");
        }
    }
}
```

Clean code

1. Descriptive Variable names & follow Naming Convention (camelCase)
2. Use comments wherever possible
3. Use proper Indentation (spaces)
4. Reduce redundancy of code

Design Principles & Design Patterns

(2) while loop

for ((I) ; (T) ; (U))

initialization before while loop

while ((T)) {

update

Exponentiation/Power :→ 2^N

$N=0 \quad 2^0 = 1$

$N=1 \quad 2^1 = 2$

$N=2 \quad 2^2 = 2*2 = 4$

$N=3 \quad 2^3 = 2*2*2 = 8$

$N=4 \quad 2^4 = 2*2*2*2 = 16$

$N=5 \quad 2^5 = 2*2*2*2*2 = 32$

int ans = 0/1/2

```
for (int idx = 1; idx <= N; idx++){
    ans = ans * 2;
}
```

N=2

$2^N = 2^2 = 4$

ans = 1

idx = 1 ≤ 2 → 1*2 = 2

idx = 2 ≤ 2 → 2*2 = 4

idx = 3 ≤ 2 ✗

```
Scanner scn = new Scanner(System.in);
int n = scn.nextInt();

int ans = 1; // 2 ^ 0 = 1

for(int idx = 1; idx <= n; idx++){
    ans = ans * 2;
}

System.out.println(ans);
```

N=5

2^5

ans

$2^0 = 1$

$2^1 = 2$

$2^2 = 4$

$2^3 = 8$

$2^4 = 16$

$2^5 = 32$

1 2 3 4 5 6

idx

1 <= 5 true : ans = 1*2 = 2

2 <= 5 true : ans = 2*2 = 4

3 <= 5 true : ans = 4*2 = 8

4 <= 5 true : ans = 8*2 = 16

5 <= 5 true : ans = 16*2 = 32

6 <= 5 false :

N = 18 ans

18 ans ✓

18/3 = 6 ans ✓

6/3 = 2 ans ✓

2/3 = 0 ans

N = 100

100 ans ✓

100/3 = 33 ans ✓

33/3 = 11 ans ✓

11/3 = 3 ans ✓

3/3 = 1 ans ✓

1/3 = 0 ans

N = 81

81 ans ✓

81/3 = 27 ans ✓

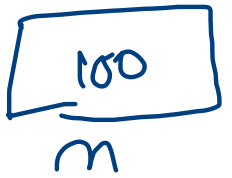
27/3 = 9 ans ✓

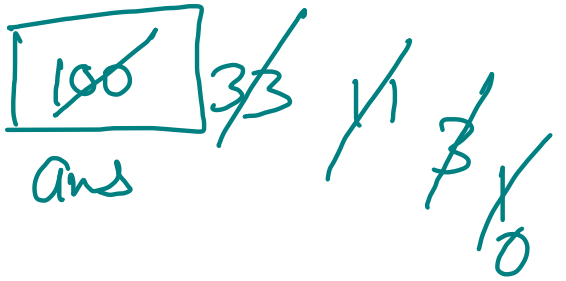
9/3 = 3 ans ✓

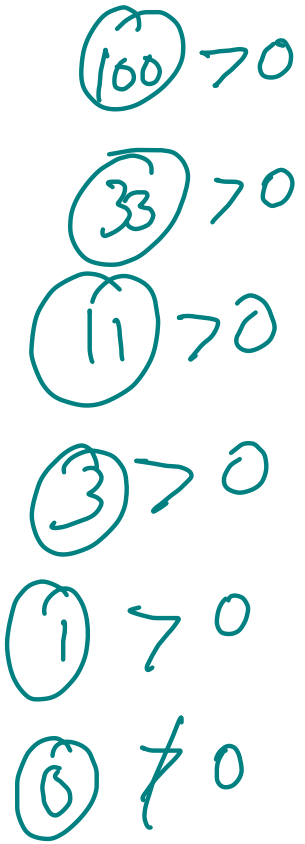
3/3 = 1 ans ✓

1/3 = 0 ans

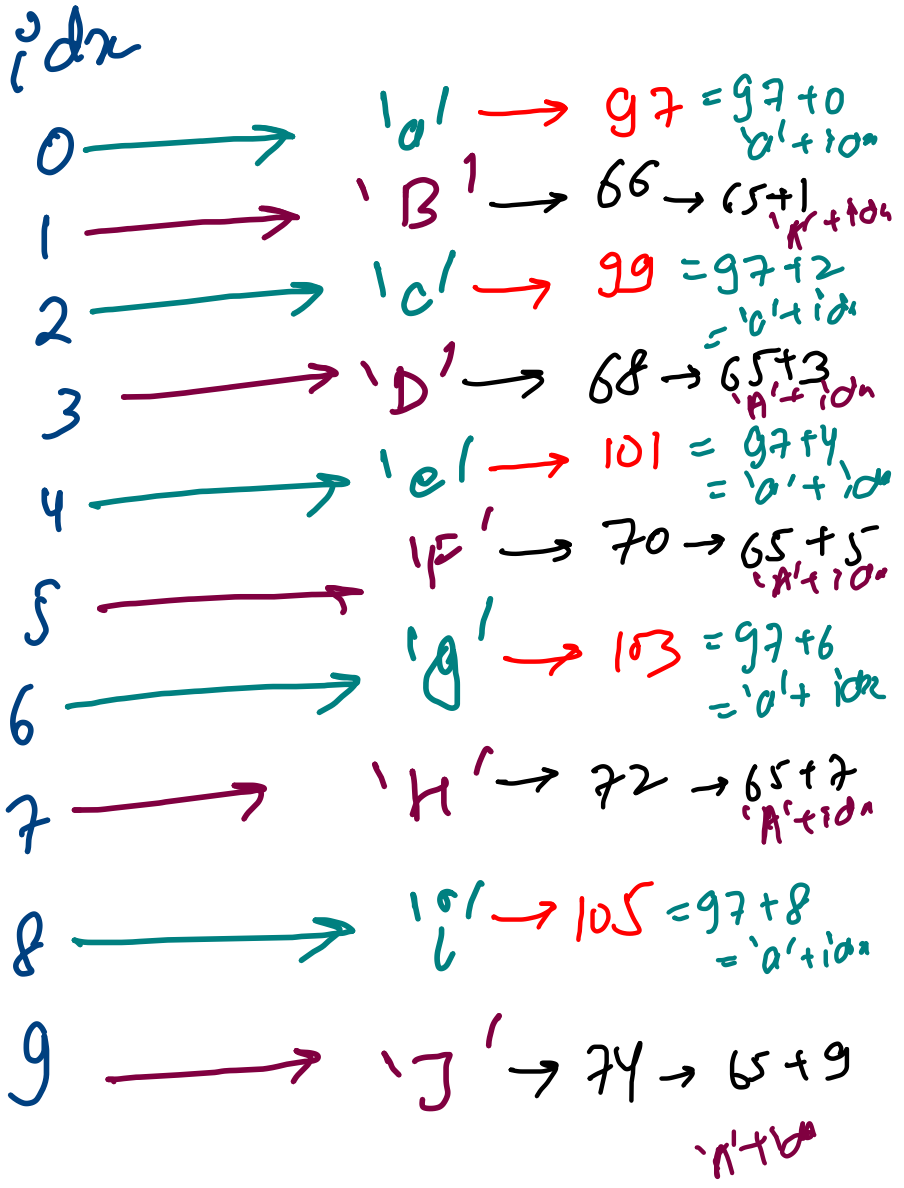
```java
Scanner scn = new Scanner(System.in);
int n = scn.nextInt();

for(int ans = n; ans > 0; ans = ans / 3){
    System.out.print(ans + " ");
}
```







```java
public static void main(String[] args) {
    for(int idx=0; idx<26; idx++){
        if(idx % 2 == 0){
            // if the idx is even -> print in lowercase
            char ch = (char)(idx + 'a');
            System.out.println(ch);
        }
        else {
            // else idx is odd -> print in uppercase
            char ch = (char)(idx + 'A');
            System.out.println(ch);
        }
    }
}
```



Soln 1

```java
public static void main(String[] args) {
    for(int idx=1; idx<=100; idx++){
        if(idx % 3 == 0 && idx % 5 == 0){
            System.out.println("FizzBuzz");
        } else if(idx % 3 == 0){
            System.out.println("Fizz");
        } else if(idx % 5 == 0){
            System.out.println("Buzz");
        } else {
            System.out.println(idx);
        }
    }
```

Soln 2

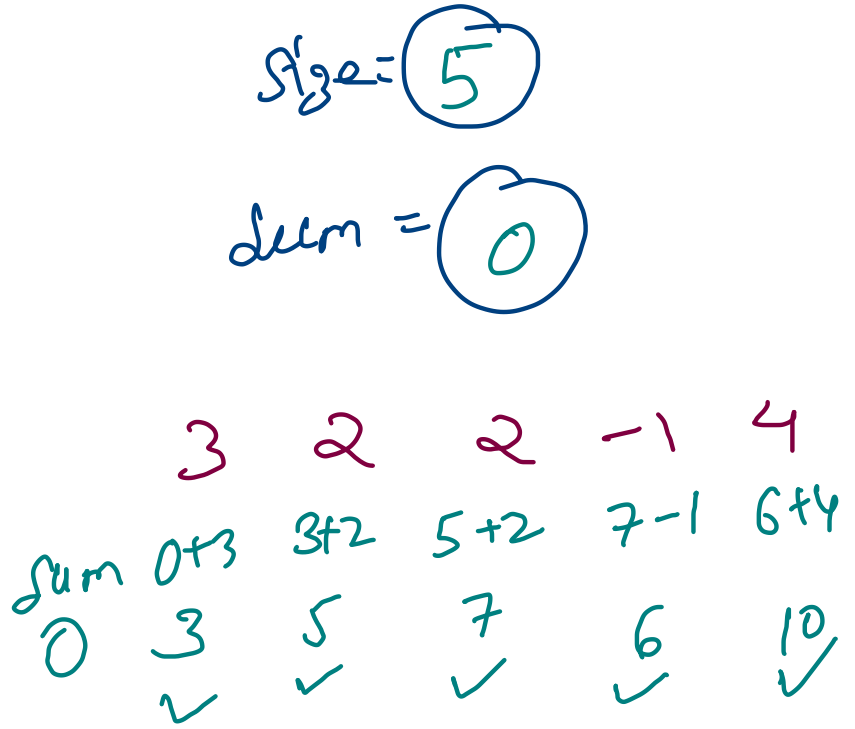
```java
public static void main(String[] args) {
    for(int idx=1; idx<=100; idx++){
        if(idx % 15 == 0){
            System.out.println("FizzBuzz");
        } else if(idx % 3 == 0){
            System.out.println("Fizz");
        } else if(idx % 5 == 0){
            System.out.println("Buzz");
        } else {
            System.out.println(idx);
        }
    }
```

Soln 3

```java
    for(int idx=1; idx<=100; idx++){
        if(idx % 3 == 0){
            System.out.print("Fizz");
        }
        if(idx % 5 == 0){
            System.out.print("Buzz");
        }
        if(idx % 3 != 0 && idx % 5 != 0){
            System.out.print(idx);
        }
        System.out.println();
    }
```

Running Sum or Prefix Sum

```
Scanner scn = new Scanner(System.in);
int size = scn.nextInt();
int sum = 0;

for(int idx=0; idx<size; idx++){
    int x = scn.nextInt();
    sum = sum + x;
    System.out.println(sum);
}
```



Print Powers of 2 < N

```java
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();

        for(int ans=1; ans<=n; ans=ans*2){
            System.out.println(ans);
        }
    }
}
```

Fibonacci Sequence

✓ ✓ ✓ ✓ ✓

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55

a b c

```
Syso(a);
```

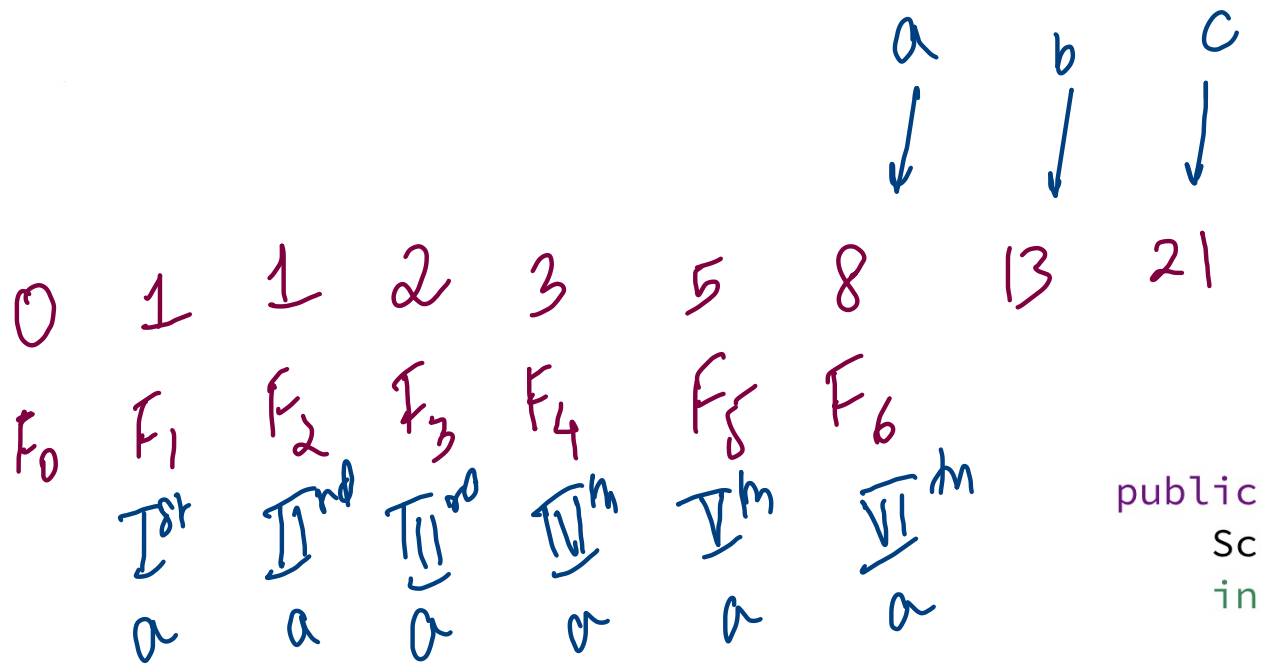
1. a = b
2. b = c
3. c = a+b

N^{th} fib = $N-1^{th}$ fib + $N-2^{th}$ fib

```java
Scanner scn = new Scanner(System.in);
int size = scn.nextInt();

int a = 0, b = 1, c = 1;

for(int idx=0; idx<size; idx++){
    System.out.print(a + " ");
    a = b;
    b = c;
    c = a + b;
}
```

Nth Fibonacci No

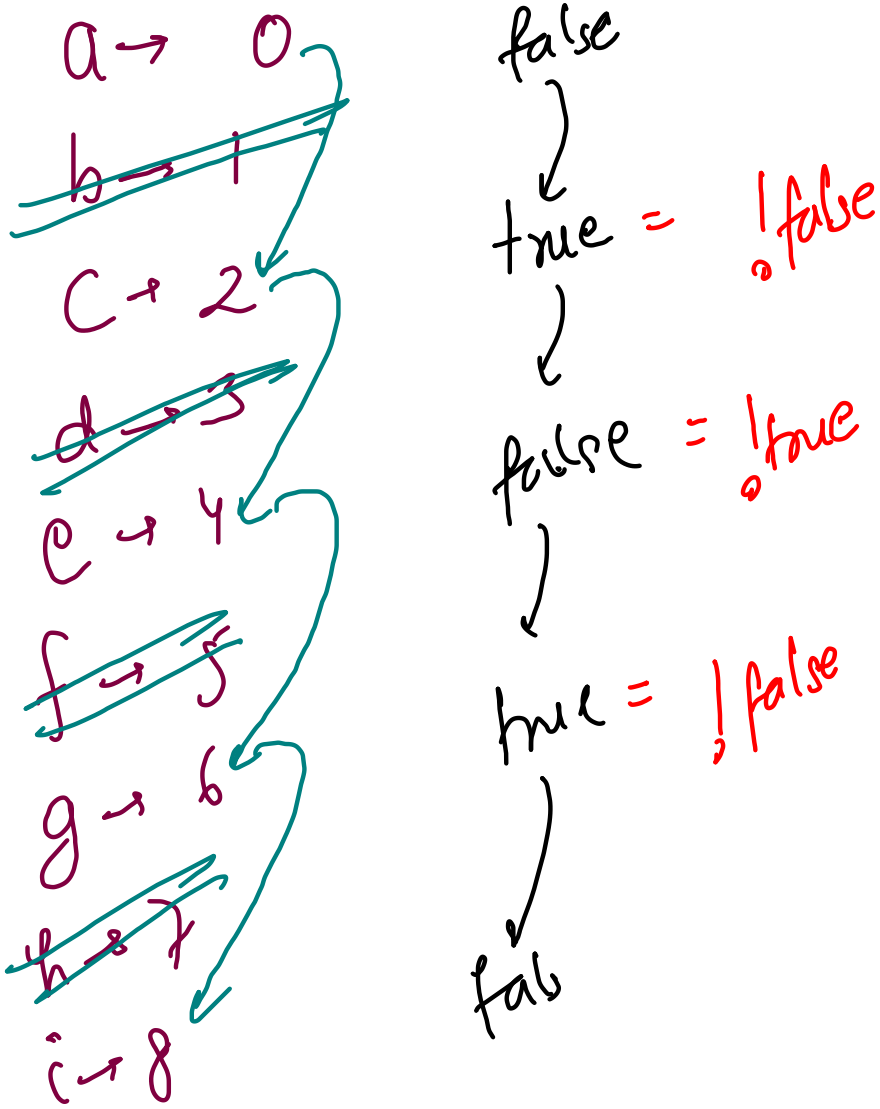
N = 6



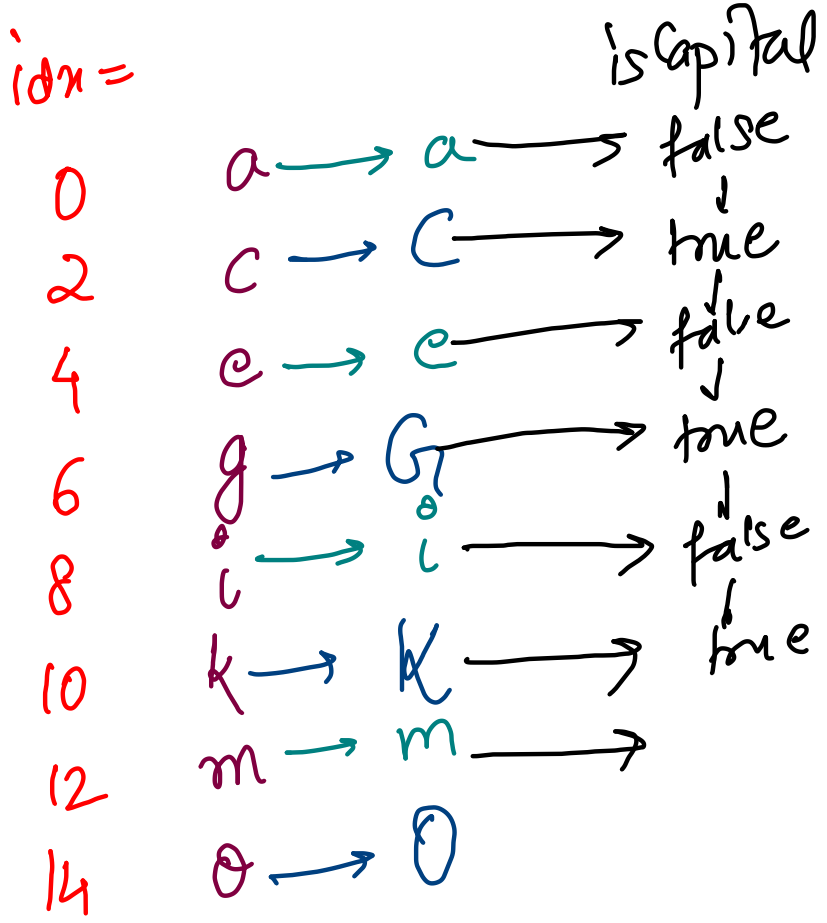

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();

    int a = 0, b = 1, c = 1;
    for(int idx=0; idx<n; idx++){
        a = b;
        b = c;
        c = a + b;
    }
    System.out.println(a);
}
```

Print a, C, e, G, i, K, ...

boolean isCapital = false

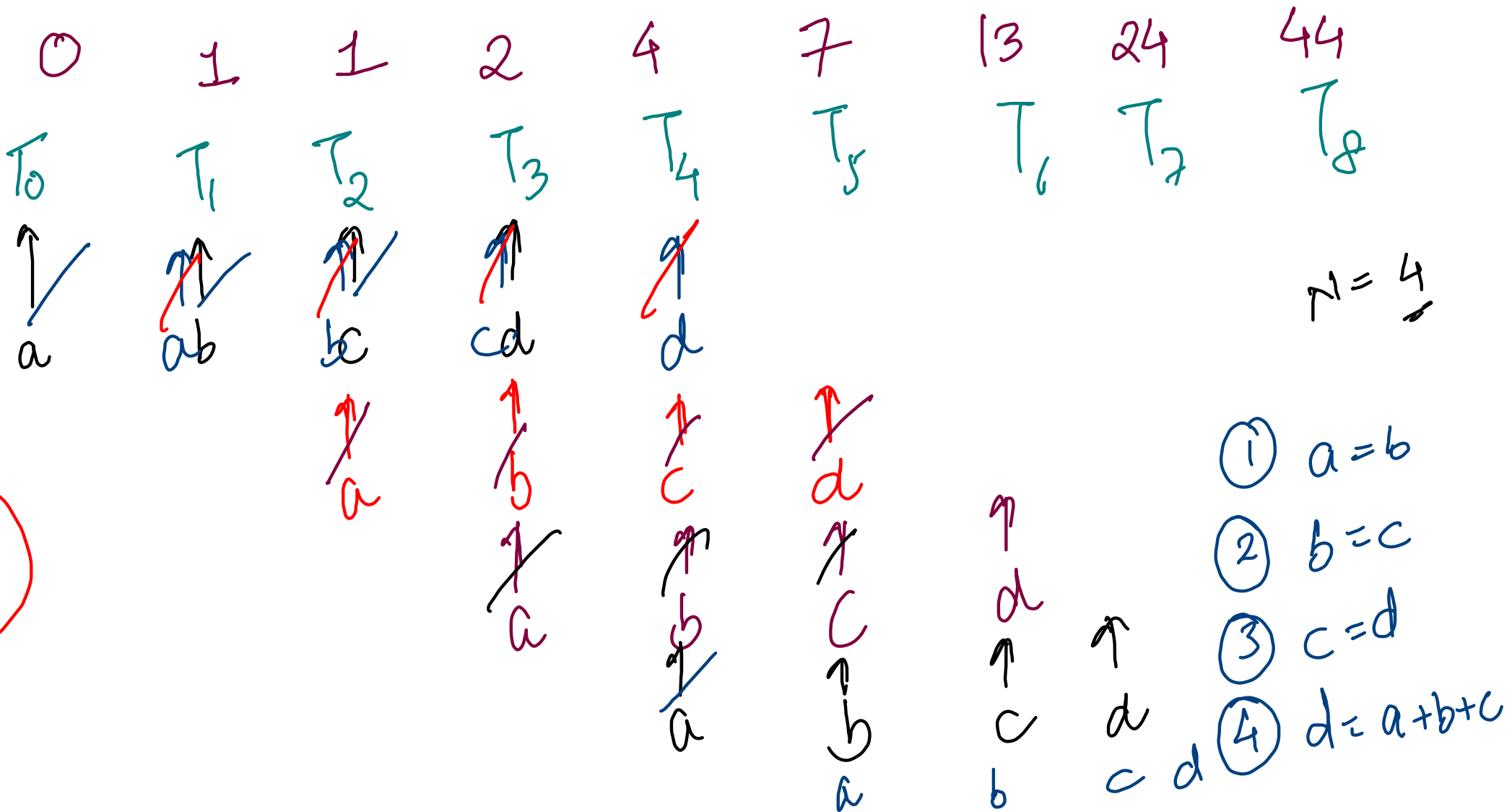


```java
public static void main(String[] args) {
    boolean isCapital = false;

    for(int idx = 0; idx < 25; idx = idx + 2){
        if(isCapital == true){
            System.out.println((char)(idx + 'A'));
            isCapital = false;
        } else {
            System.out.println((char)(idx + 'a'));
            isCapital = true;
        }
    }
}
```



```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int steps = 0;

    while(n > 0){
        if(n % 2 == 0){
            n = n - 1;
        } else {
            n = n - 3;
        }
        steps++;
    }

    System.out.println(steps);
}
```

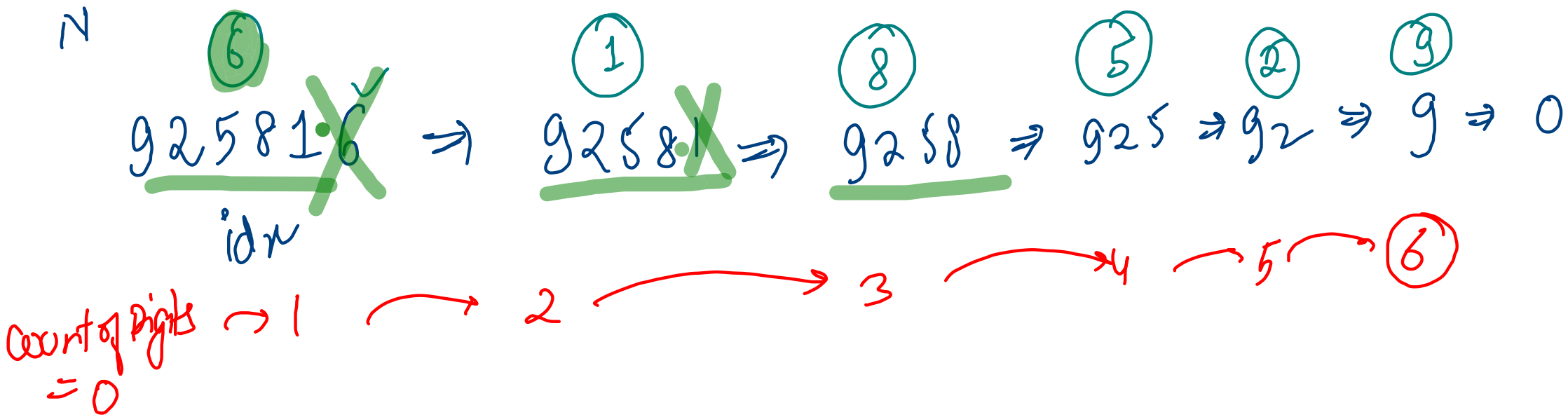
Tribonacci sequence



```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();

    int a = 0, b = 1, c = 1, d = 2;
    for(int idx=0; idx<n; idx++){
        a = b;
        b = c;
        c = d;
        d = a + b + c;
    }
    System.out.println(a);
}
```

```java
Scanner scn = new Scanner(System.in);
int n = scn.nextInt();

for(int idx=n; idx>0; idx=idx/10){
    System.out.println(idx % 10);
}
```

Extracting digits from a No

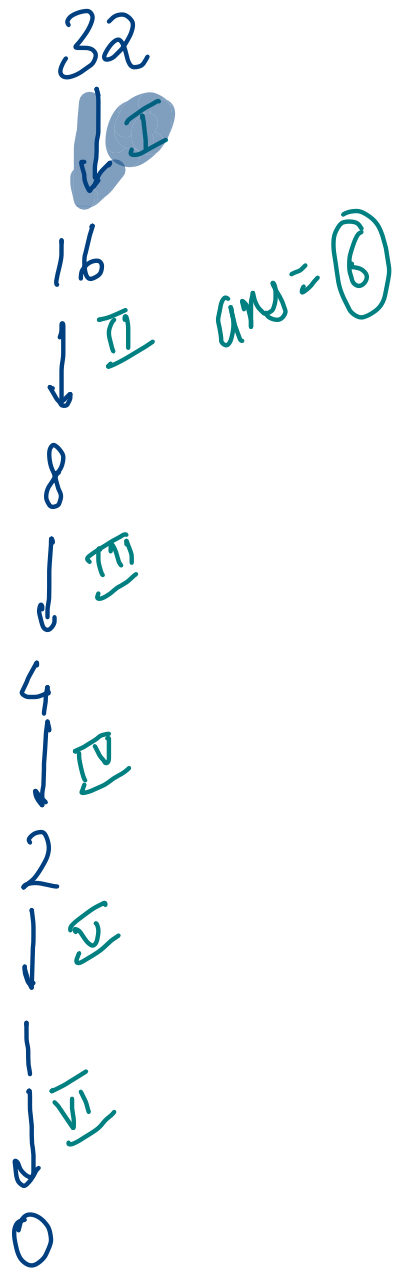


```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();

    int countOfDigits = 0;
    for(int idx = n; idx > 0; idx = idx / 10){
        countOfDigits++;
    }

    System.out.println(countOfDigits);
}
```

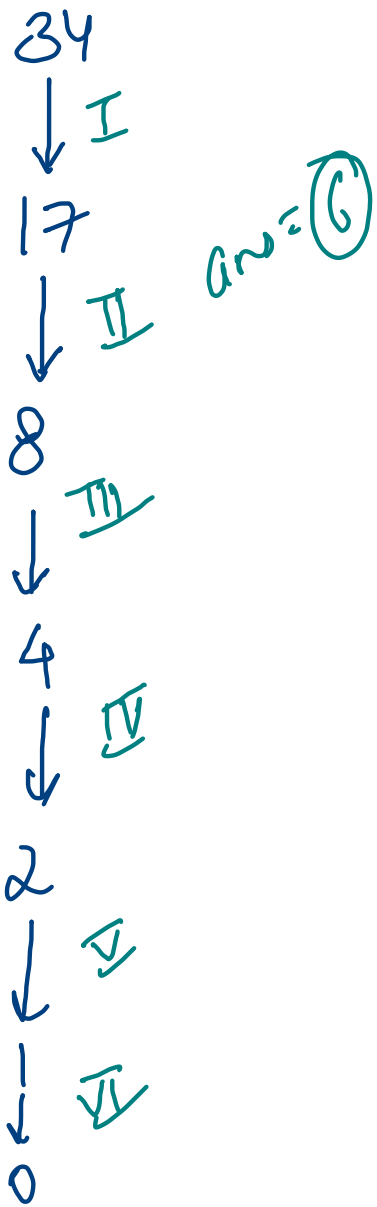
n = 61592437

idx	count of digits
61592437	0
6159243	1
615924	2
61592	3
6159	4
615	5
61	6
6	7
0	8

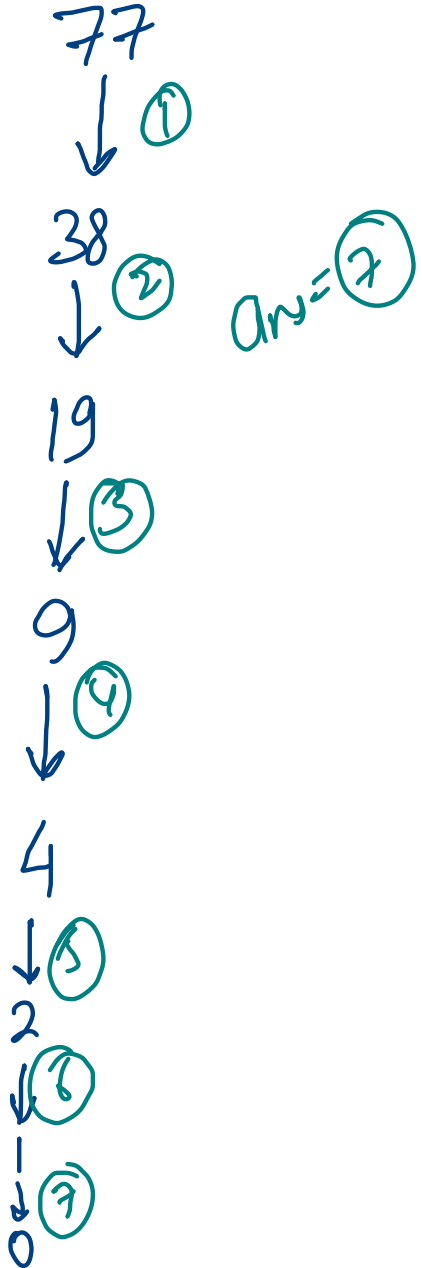
No of steps on N/2



ans = 6



ans = 6



ans = 7

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int countOfSteps = 0;

    for(int idx = n ; idx > 0; idx = idx / 2){
        countOfSteps++;
    }

    System.out.println(countOfSteps);
}
```

size → 8	max	steps
	~~-100~~	~~0~~
5	5	1
3	~~5~~	~~1~~
7	7	2
7	~~7~~	~~2~~
9	9	3
2	~~9~~	~~3~~
10	10	4
10	10	(4)

```java
Scanner scn = new Scanner(System.in);
int size = scn.nextInt();

int max = -100, steps = 0;

for(int idx=0; idx<size; idx++){
    int x = scn.nextInt();
    if(x > max){
        max = x;
        steps++;
    }
}

System.out.println(steps);
```

get how many inputs we require

max at any instant

counter to check how many times you have updated the max variable

to take (size) no of inputs

Size = 5

idx	x
0	3
1	1
2	5
3	5
4	9

max =
~~-100~~
3
~~3~~
5
~~5~~
9

steps
~~0~~
1
~~1~~
2
~~2~~
3

print

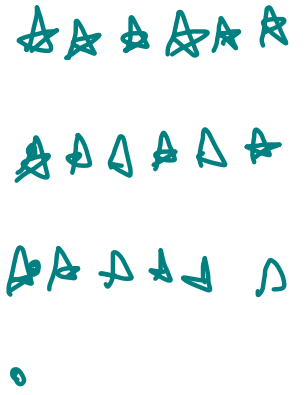
Rows = 7

Columns = 6

```
for(int row = 0; row<3; row++){
    for(int col=0; col< 6; col++){
        SOP("*");
    }
    System.out.println();
}
```

row = 0 1 2

col = 0 -> 5



```java
for(int row=0; row<6; row++){
    for(int col=0; col<4; col++){
        System.out.println("row " + row + " col " + col);
    }
    System.out.println();
}
```

row = 0 col = 0
row = 0 col = 1
row = 0 col = 2
row = 0 col = 3
row = 1 col = 0
row = 1 col = 1
row = 1 col = 2
row = 1 col = 3
row = 2 col = 0
row = 2 col = 1
row = 2 col = 2
row = 2 col = 3

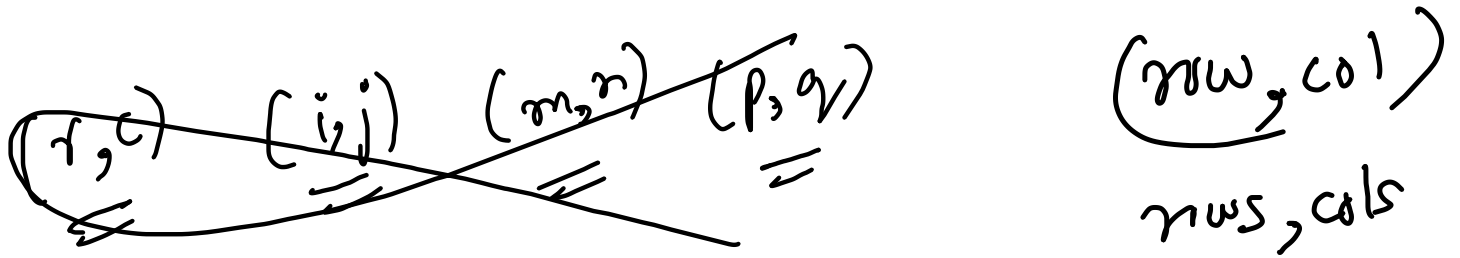
~~(4)~~

row 0 col 0
row 0 col 1
row 0 col 2
row 0 col 3
row 1 col 0
row 1 col 1
row 1 col 2
row 1 col 3

(13)

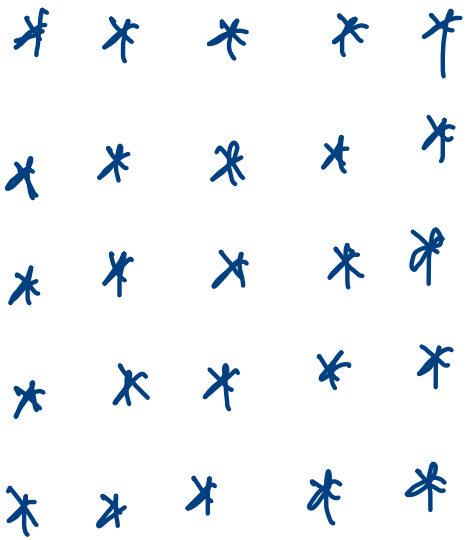
row 0 col 0
row 1 col 0
row 2 col 0
row 0 col 1
row 1 col 1
row 2 col 1
row 0 col 2
row 1 col 2
row 2 col 2

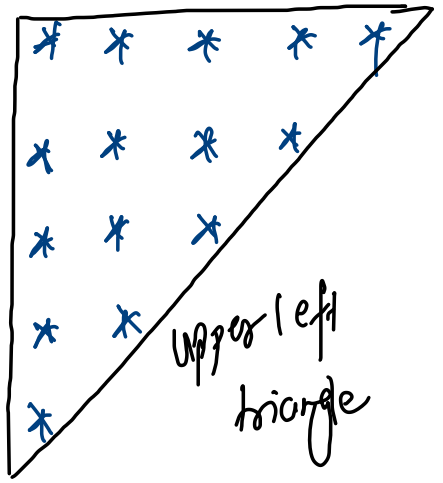
```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int cols = scn.nextInt();

    for(int row=0; row<12; row++){
        for(int col=0; col<cols; col++){
            System.out.print("*");
        }
        System.out.println();
    }
}
```

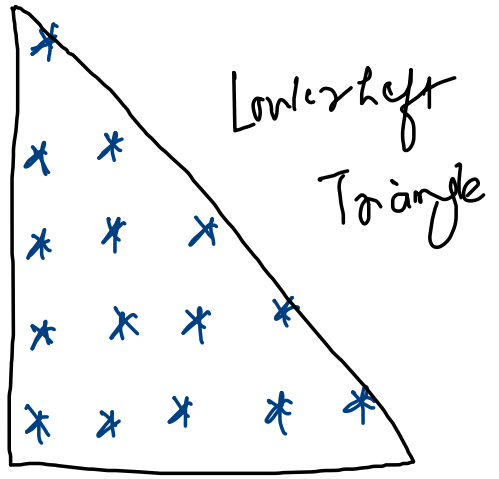


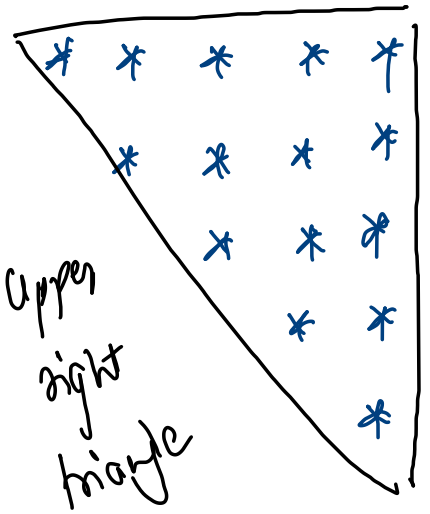
```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();

    int rows = n, cols = n;
    for(int row=0; row<rows; row++){
        for(int col=0; col<cols; col++){
            System.out.print("*");
        }
        System.out.println();
    }
}
```

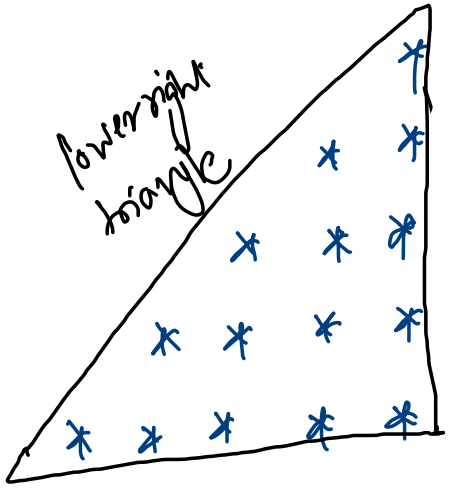
5 x 5



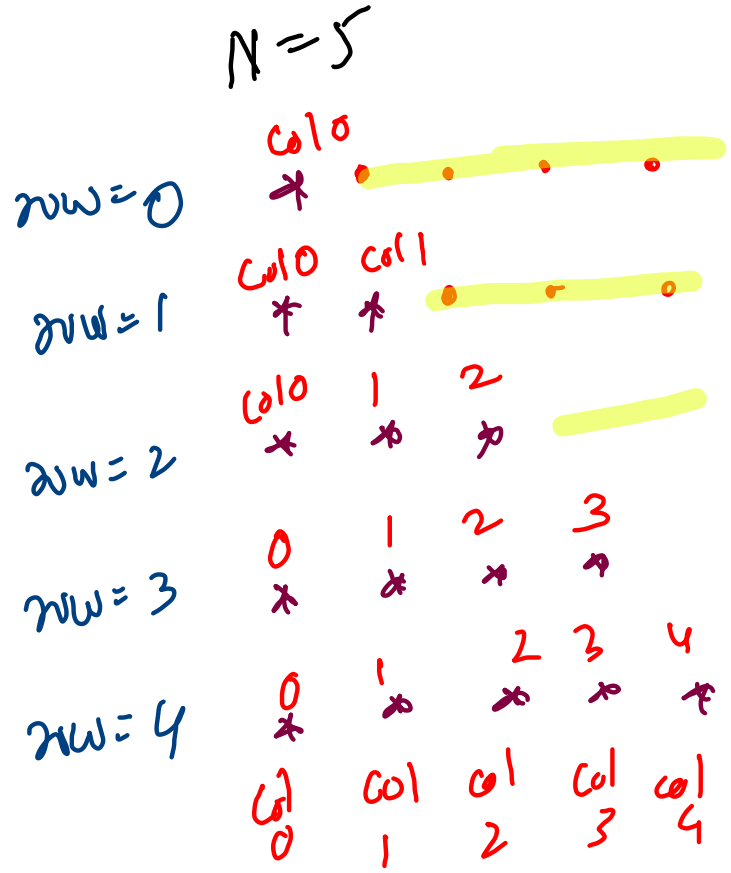

Upper left
triangle


Lower left
Triangle

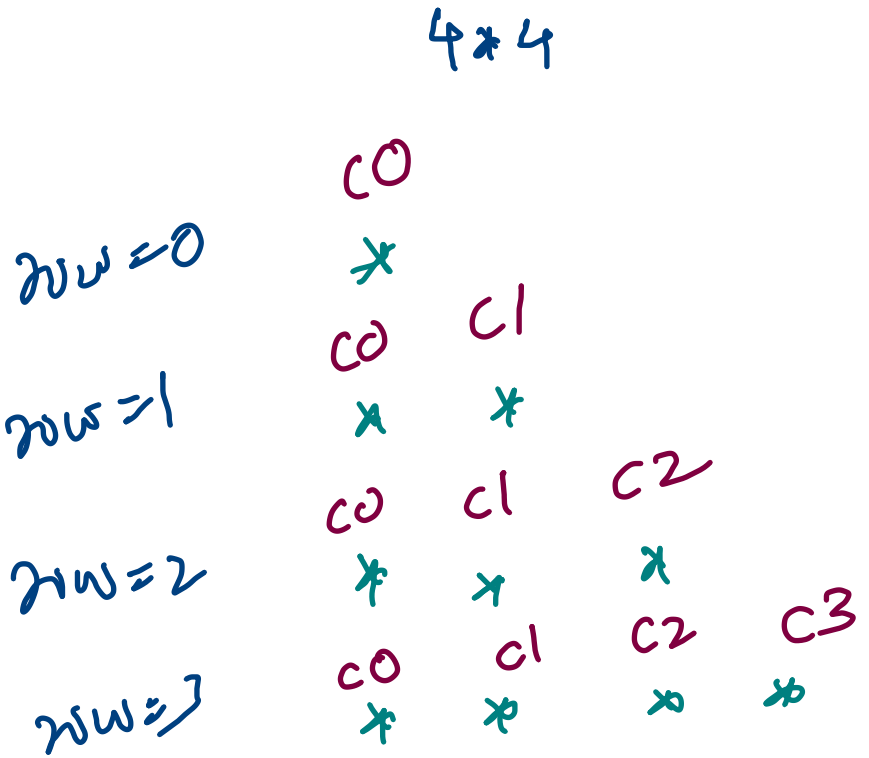

Upper
right
triangle

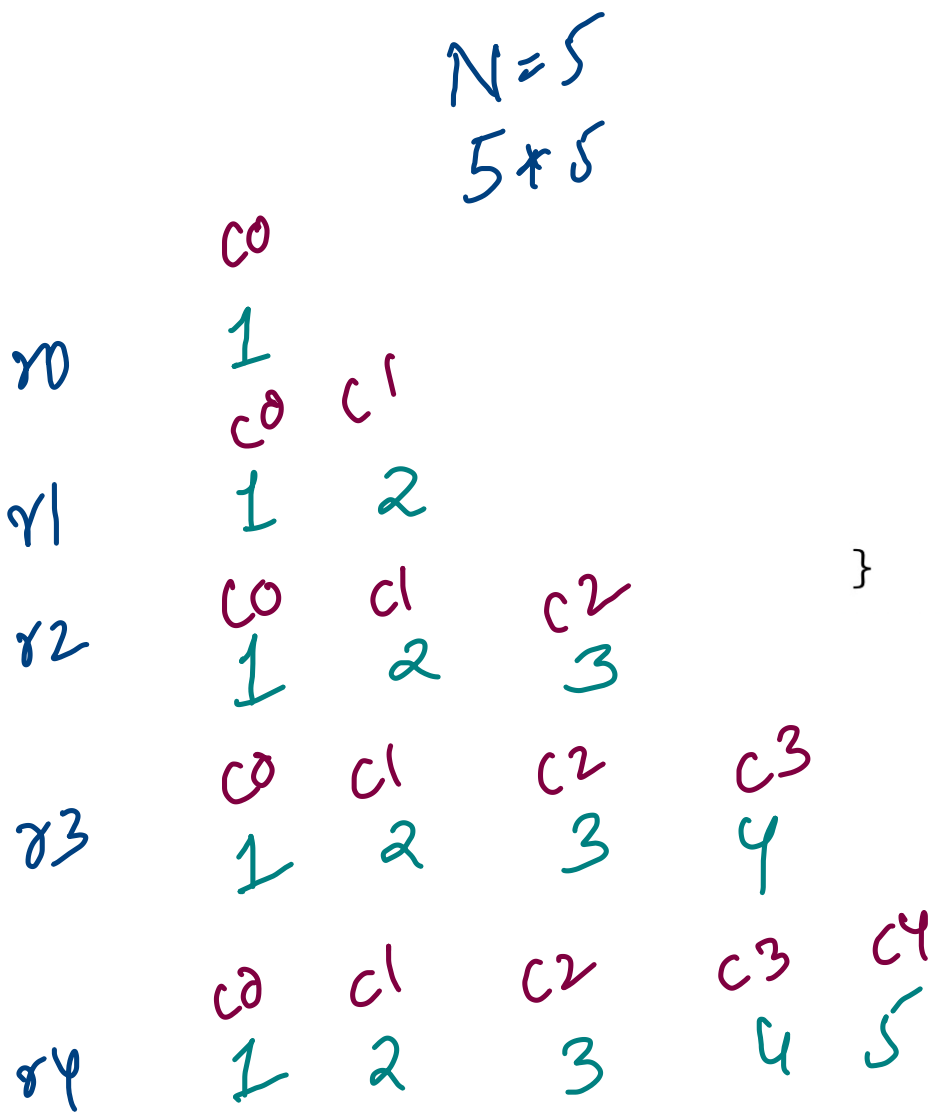

lower right
triangle

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();

    int rows = n, cols = n;
    for(int row = 0; row < rows; row++){
        for(int col = ?; col < ?; col++){
            System.out.print("*");
        }
        System.out.println();
    }
}
```



```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();

    int rows = n, cols = n;
    for(int row = 0; row < rows; row++){
        for(int col = 0; col <= row; col++){
            System.out.print("* ");
        }
        System.out.println();
    }
}
```






```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();

    int rows = n, cols = n;
    for(int row = 0; row < rows; row++){
        for(int col = 0; col <= row; col++){
            System.out.print(col + 1 + " ");
        }
        System.out.println();
    }
}
```

[5, 14]

① excluding 5 & 14

(8)

$r-l-1$

② only including 5

(9)

$r-l$

③ only including 14

(9)

$r-l$

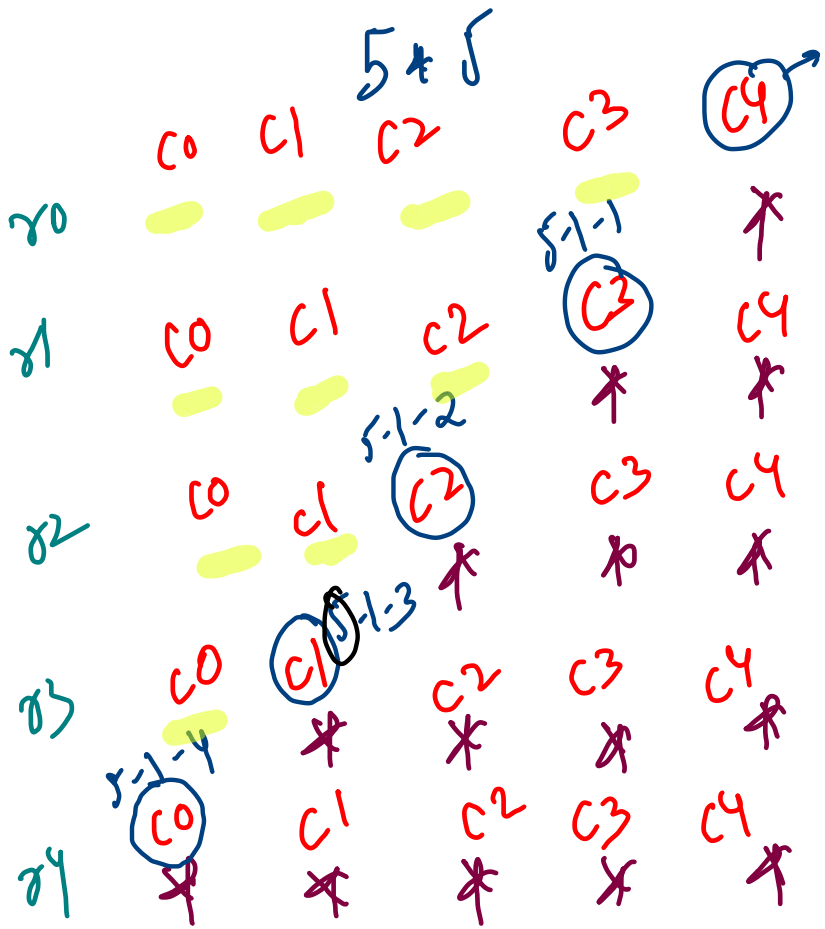
④ including 5 & 14

(10)

$r-l+1$

Lower Right Triangle

```
for(col=n-1 ; col>=n-1-row ; col--)
```



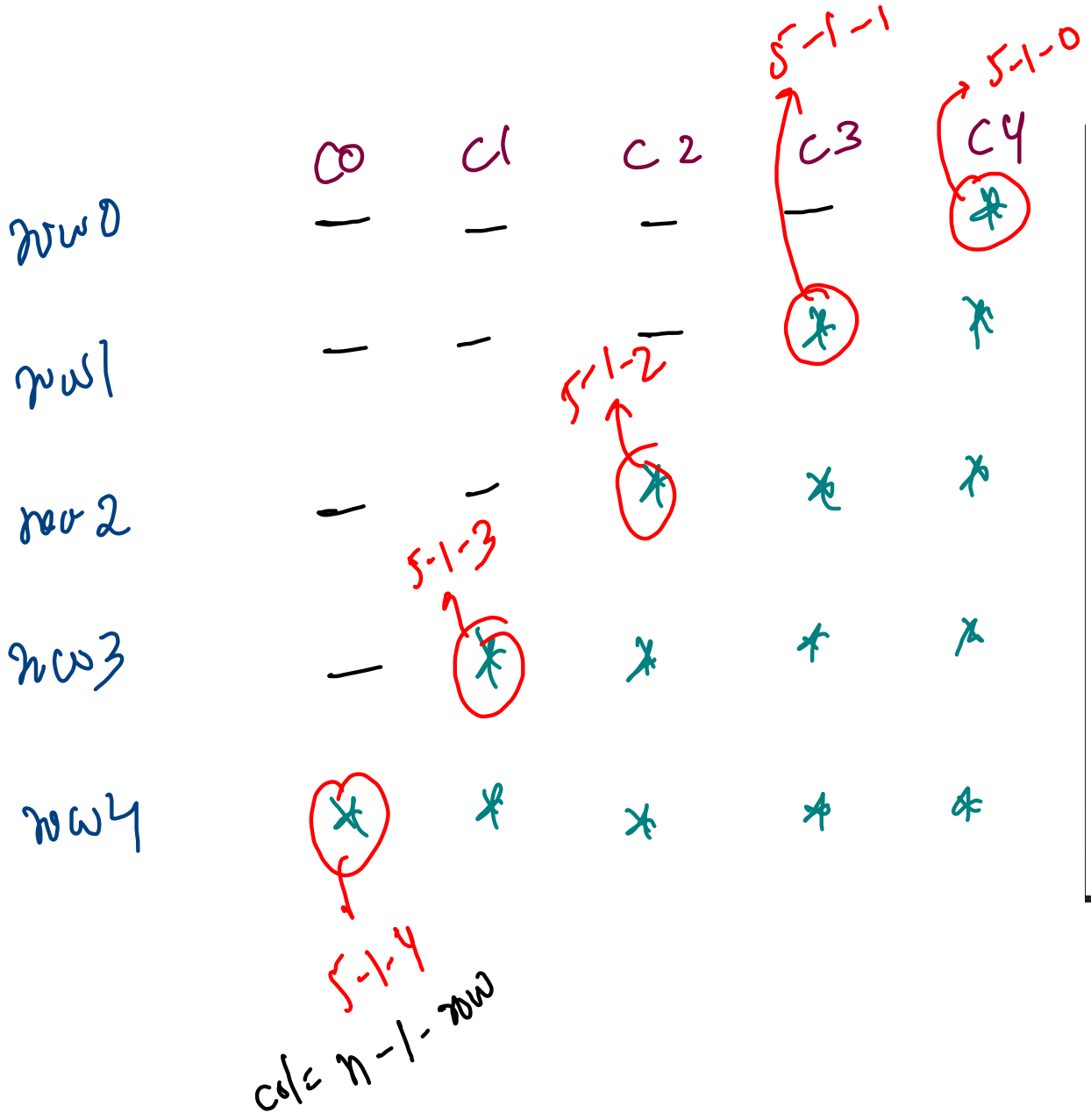
col

- 4
- 4 → 3
- 4 → 3 → 2
- 4 → 3 → 2 → 1
- 4 → 3 → 2 → 1 → 0

Reversing the loop for column will reverse the pattern

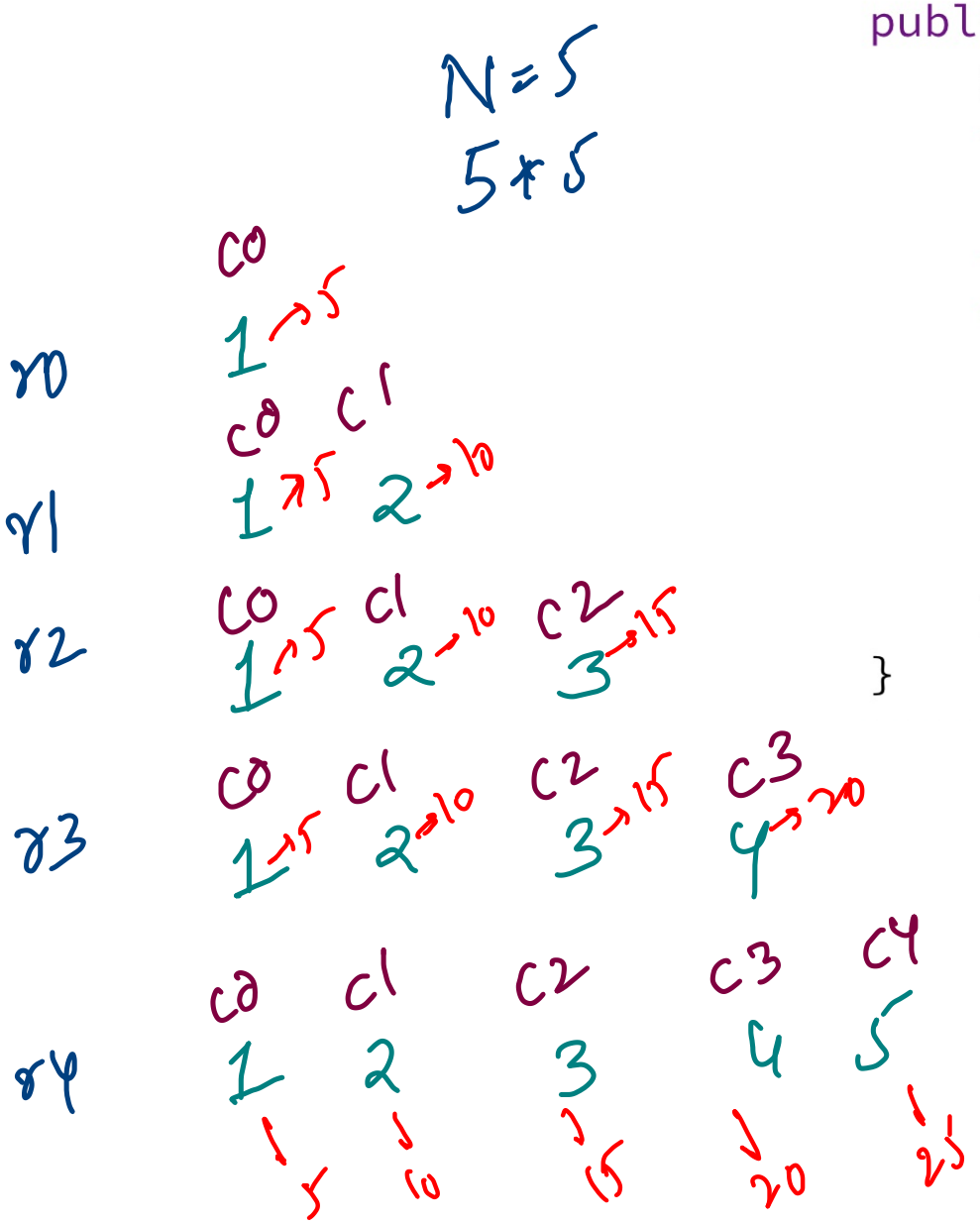
5*5

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int rows = n, cols = n;
    for(int row = 0; row < rows; row++){
        for(int col = cols - 1; col >= cols - 1 - row; col--){
            System.out.print("*");
        }
        System.out.println();
    }
}
```




```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int rows = n, cols = n;
    for(int row = 0; row < rows; row++){
        for(int col = 0; col < cols; col++){
            if(col < n - 1 - row){
                // Space
                System.out.print(" ");
            } else {
                // Star
                System.out.print("*");
            }
        }
        System.out.println();
    }
}
```

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int rows = n, cols = n;
    for(int row = 0; row < rows; row++){
        for(int col = 0; col < cols; col++){
            if(col < n - 1 - row){
                // Space
                System.out.print(" ");
            } else {
                // Star
                System.out.print("*");
            }
        }
        System.out.println();
    }
}
```

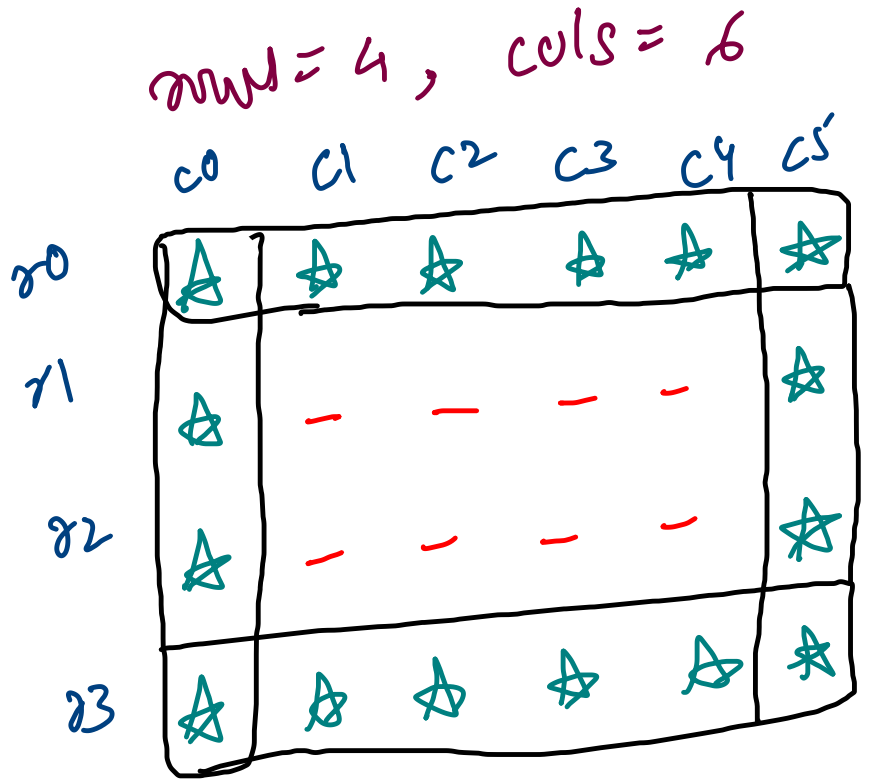



```
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();

    int rows = n, cols = n;
    for(int row = 0; row < rows; row++){
        for(int col = 0; col <= row; col++){
            System.out.print(col + 1 + " ");
        }
        System.out.println();
    }
}
```

```java
Scanner scn = new Scanner(System.in);
int cols = scn.nextInt();
int rows = scn.nextInt();

for(int row = 0; row < rows; row++){
    for(int col = 0; col < cols; col++){
        if(??){
            System.out.print("*");
        } else {
            System.out.print(" ");
        }
    }
    System.out.println();
}
```

→ Boundary

→ Inside



if(row == 0 || col == 0 || row == rows - 1 || col == cols - 1)

Top wall — Left wall — Bottom wall — Right wall

→ Star → Boundaries

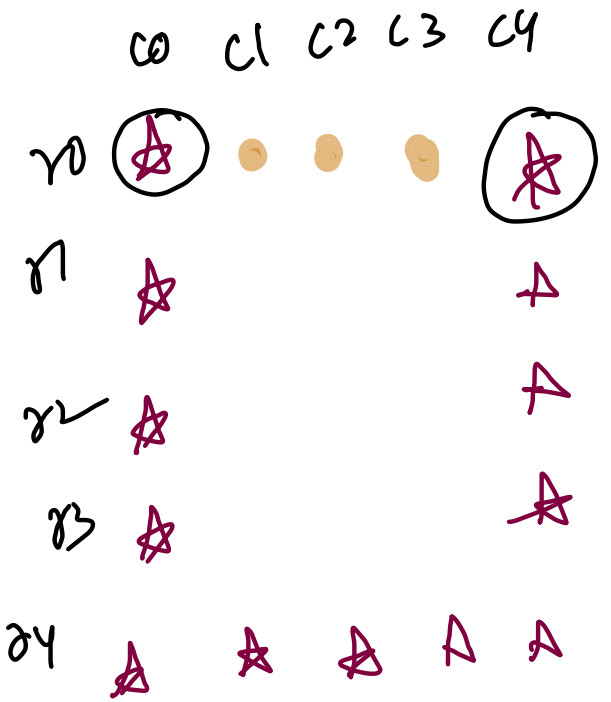
```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();

    int rows = n, cols = n;
    for(int row = 0; row < rows; row++){
        for(int col = 0; col < cols; col++){
            if(col == 0 || row == rows - 1 || col == cols - 1){
                System.out.print("*");
            } else {
                System.out.print(" ");
            }
        }
        System.out.println();
    }
}
```

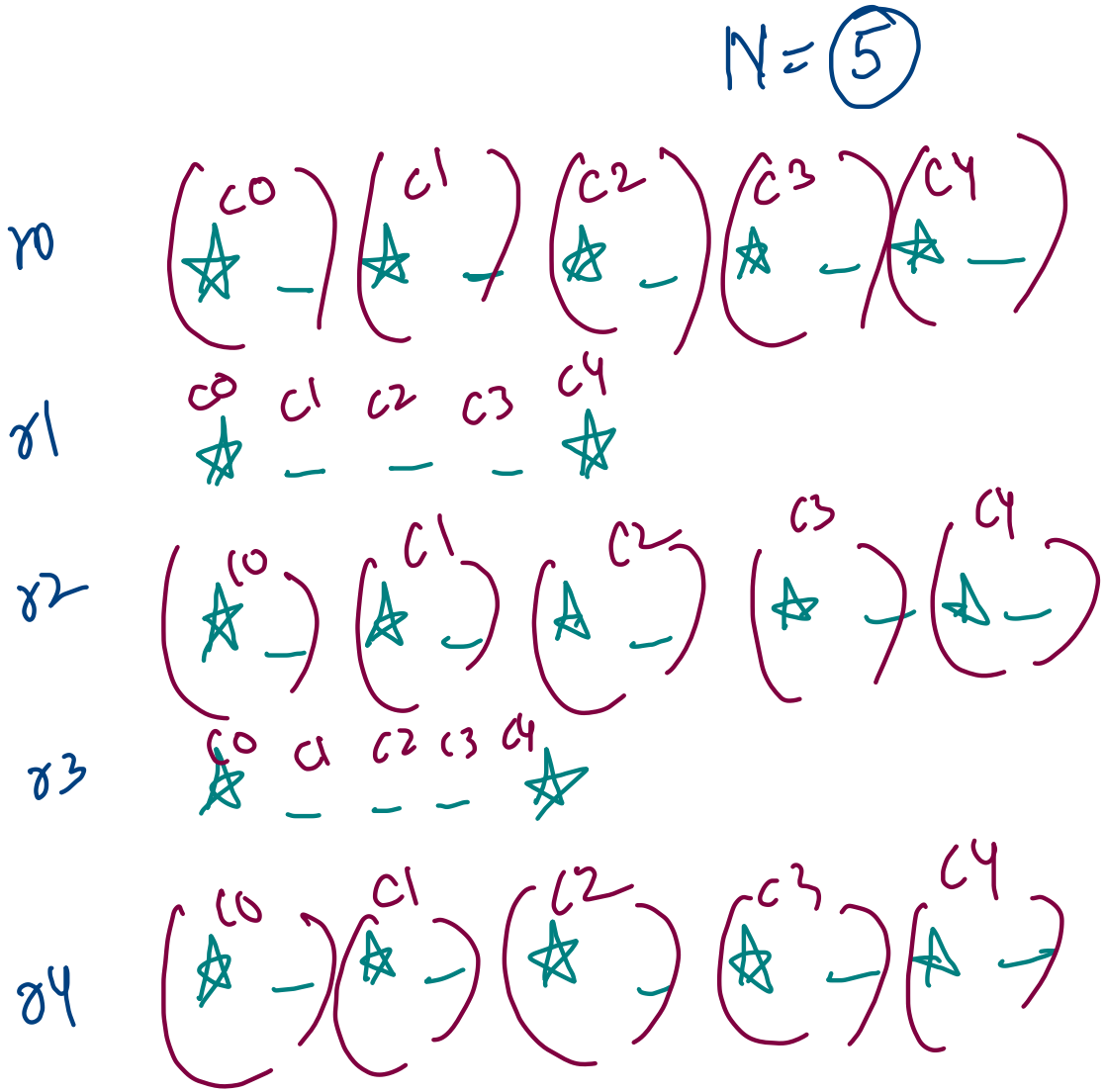
leftwall

Bottom wall

Right wall

N = 5

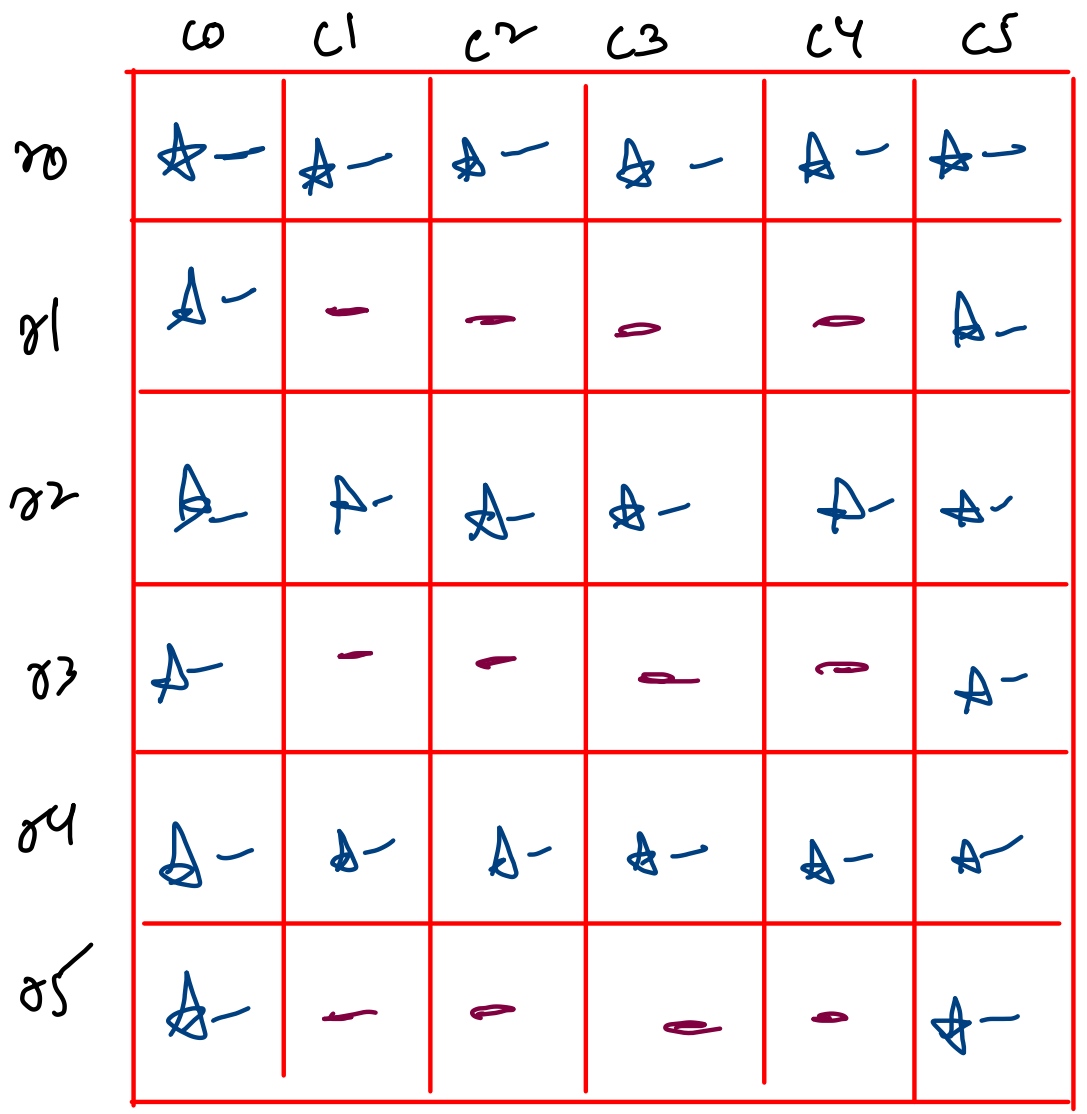





```java
public class Solution {
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int rows = n, cols = n;

        for(int row = 0; row < rows; row++){
            for(int col = 0; col < cols; col++){
                if(??){
                    System.out.print("* ");
                }
                else
                {
                    if(??){
                        System.out.println("*");
                    } else {
                        System.out.println(" ");
                    }
                }
            }
        }
    }
}
```

```java
import java.io.*;
import java.util.*;

public class Solution {
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int rows = n, cols = n;

        for(int row = 0; row < rows; row++){
            for(int col = 0; col < cols; col++){
                if(row % 2 == 0 || col == 0 || col == cols - 1){
                    System.out.print("*\t");
                }
                else {
                    System.out.print("\t");
                }
            }
            System.out.println();
        }
    }
}
```

N = 6



tab, newline, backspace, return

\t \n \b \r

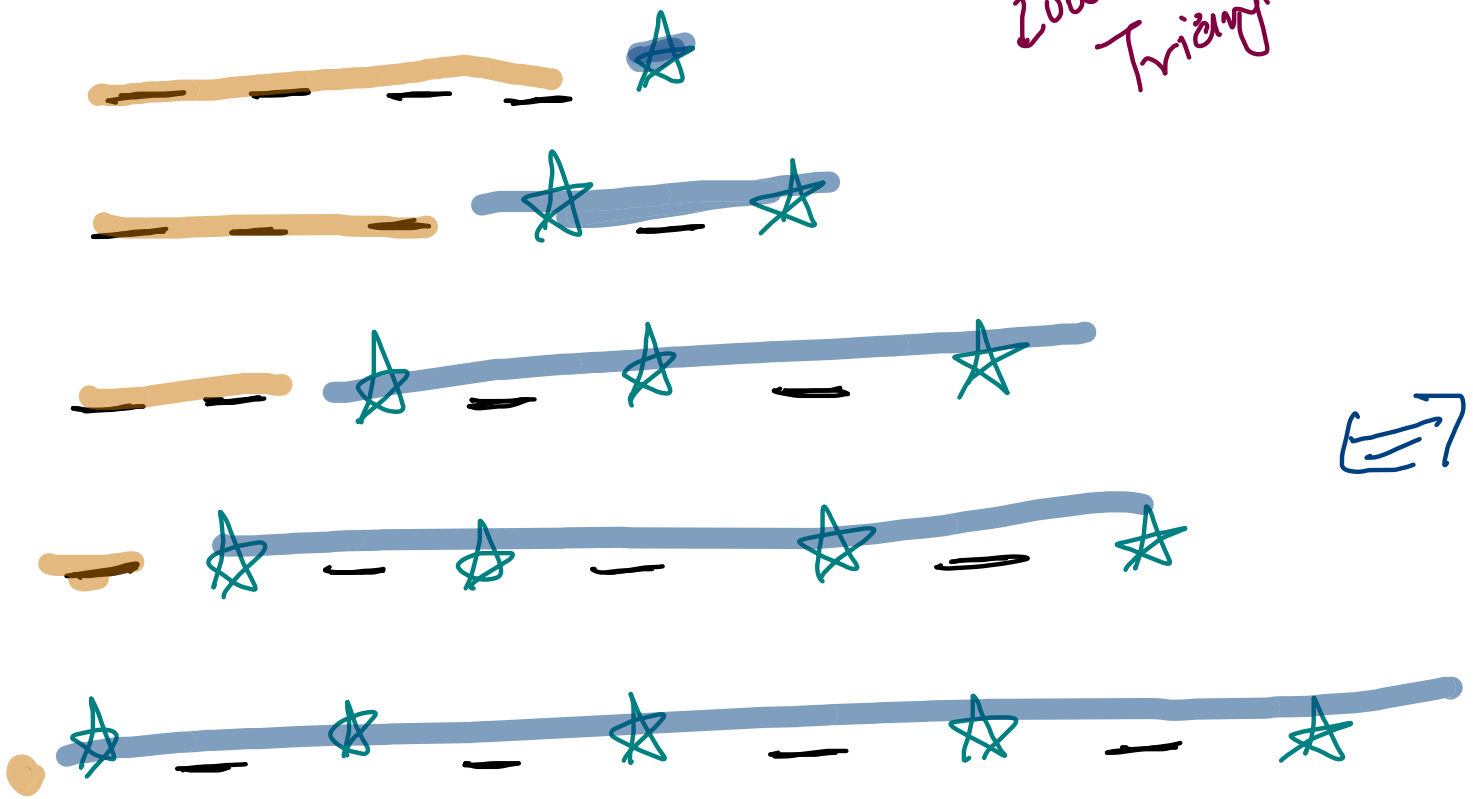
backslash + character : escape character

Pyramid

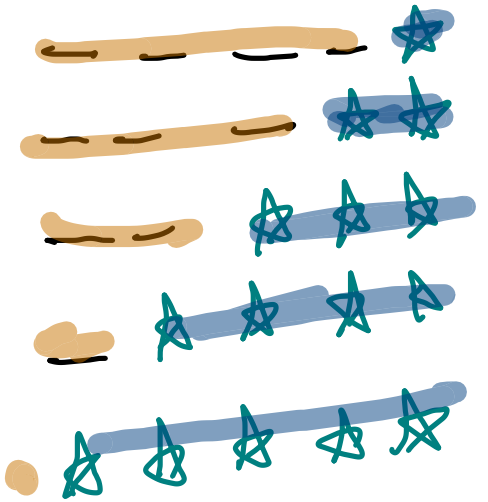
Pyramid

NEJ

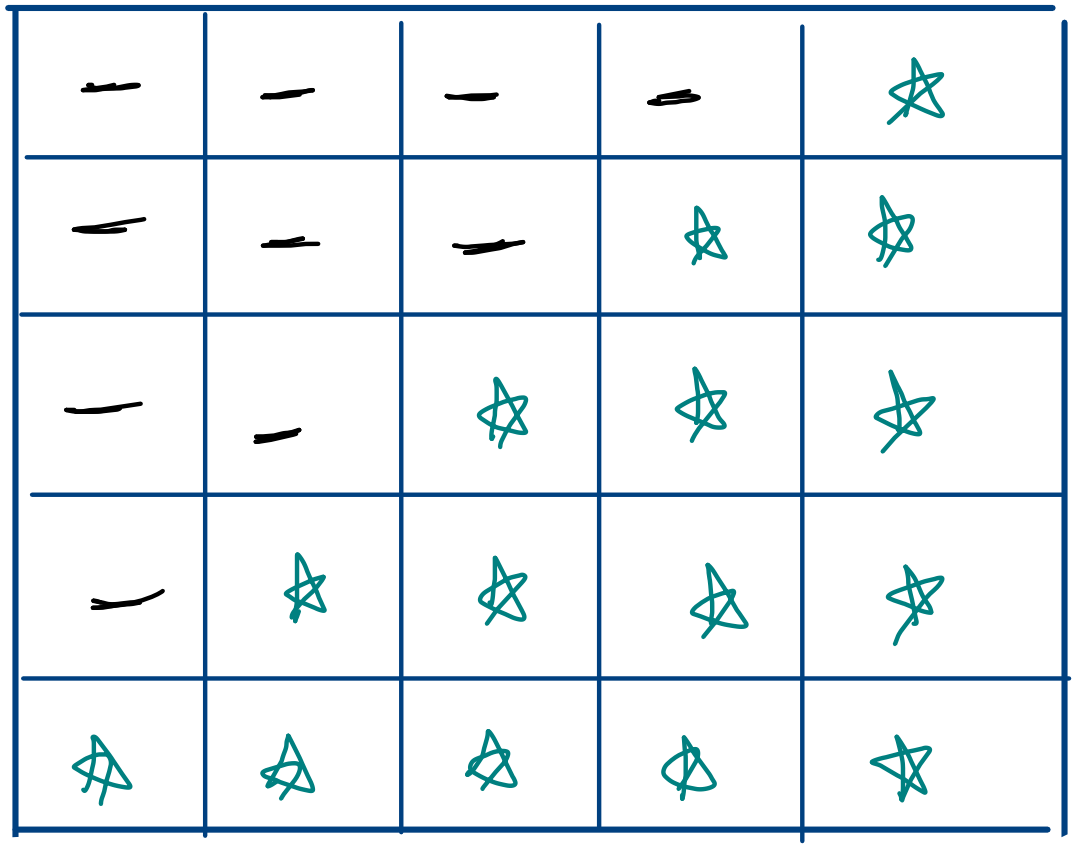
Hint :-> Lower Right Triangle



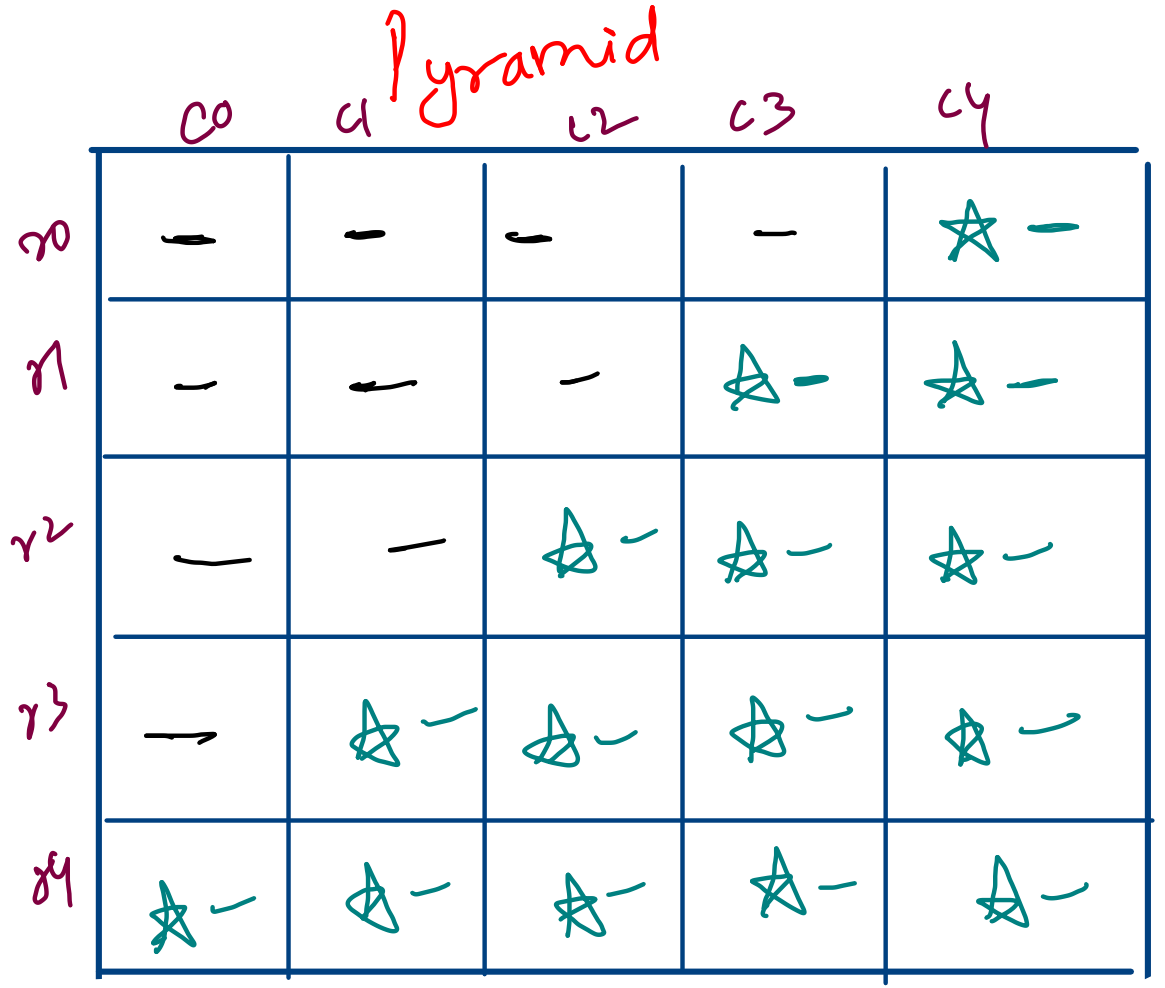
lower Right Triangle



Lower Right Triangle



Pyramid



N=5

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int rows = n, cols = n;
    for(int row = 0; row < rows; row++){
        for(int col = 0; col < cols; col++){
            if(col < n - 1 - row){
                // Space
                System.out.print(" ");
            } else {
                // Star
                System.out.print("* ");
            }
        }
        System.out.println();
    }
}
```

Function (need)

1. Reduce the code redundancy
2. Code Reusability is increased
3. Code Reability/maintability is improved
4. Debugging the code becomes easier

returntype

```
public static void greeting () {
        System.out.println( "Namaste");
}

public static void main( String [] args) {
        greeting();
}
```

- no input → parameters
- no output → return type

```java
import java.io.*;
import java.util.*;

public class Solution {
    public static void sum(int x, int y){
        System.out.println(x + y);
    }

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int x = scn.nextInt();
        int y = scn.nextInt();
        sum(x, y);
    }
}
```

function code

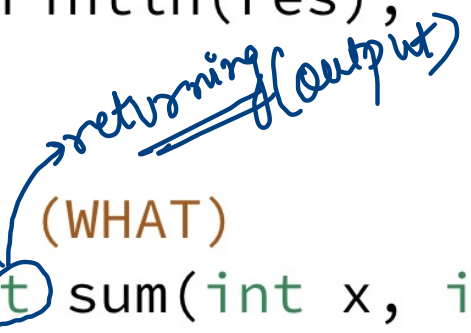
call the function

- Input ✓
- Output ✓

```
public static int sum(int a, int b){
        int res = a+b;
        return res;
}

public static void main(String[] args){
        int a=5, b=10;
        int res = sum(a,b);
        Syso(res);
}
```

- Parameters ✓
- Return type ✓

```java
public class Solution {
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int x = scn.nextInt();
        int y = scn.nextInt();
        int res = sum(x, y); // function call (WHEN)
        System.out.println(res);
    }

    // function body (WHAT)
    public static int sum(int x, int y){
        int res = x + y;
        return res;
    }
}
```

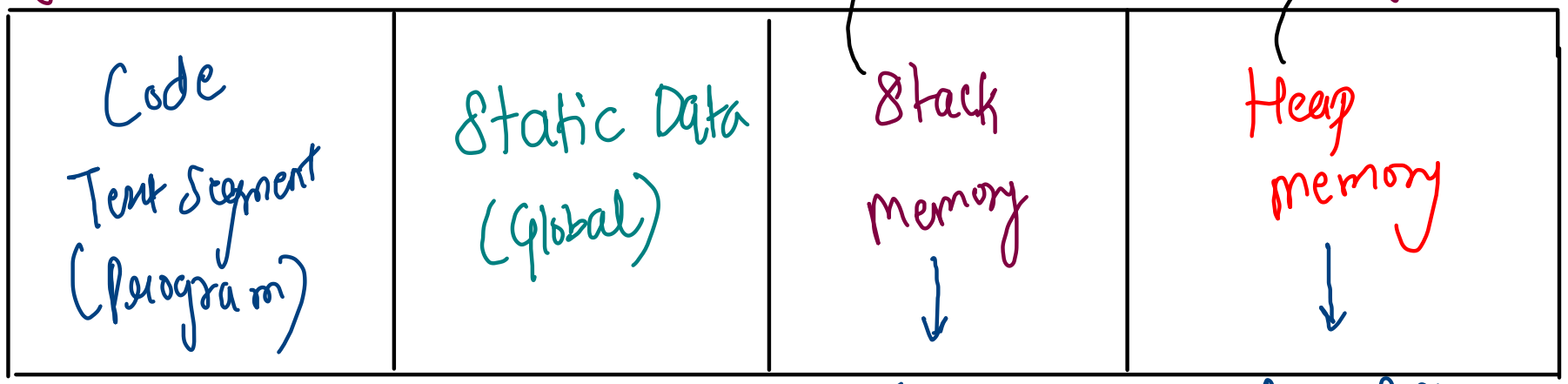


Program vs Process

Program → set of instructions

Process → Program under execution
under CPU, RAM, HDD/SSD

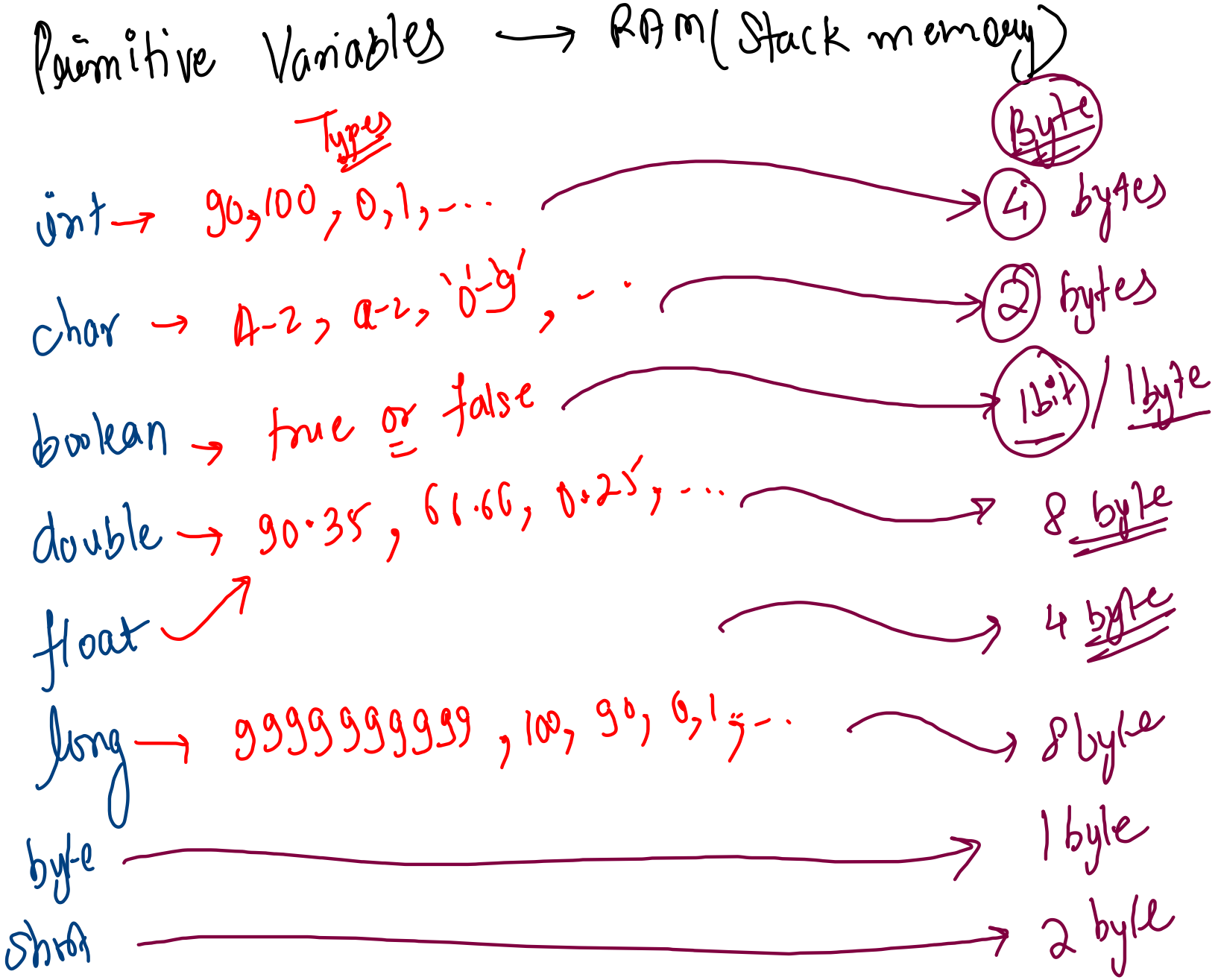
Process memory layout



RAM

- Primitive
- Reference

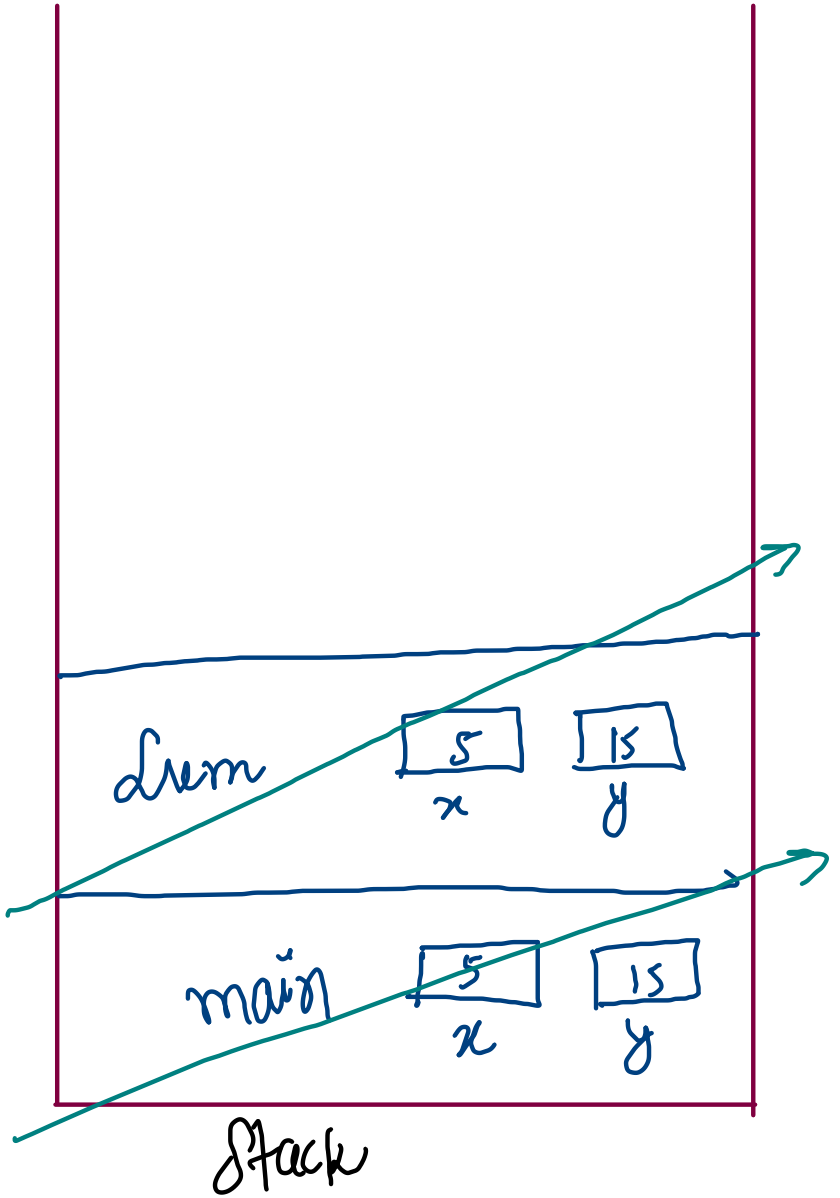
- Arrays & Strings
- Objects
  - Collection Framework
  - UserDefined memory


Primitive Variables → RAM (Stack memory)
Types
Byte
int → 90, 100, 0, 1, ...
4 bytes
char → A-Z, a-z, '0-9', ...
2 bytes
boolean → true or false
1bit / 1byte
double → 90.35, 61.66, 0.25, ...
8 byte
float
4 byte
long → 9999999999, 100, 90, 0, 1, ...
8byte
byte
1byte
short
2 byte

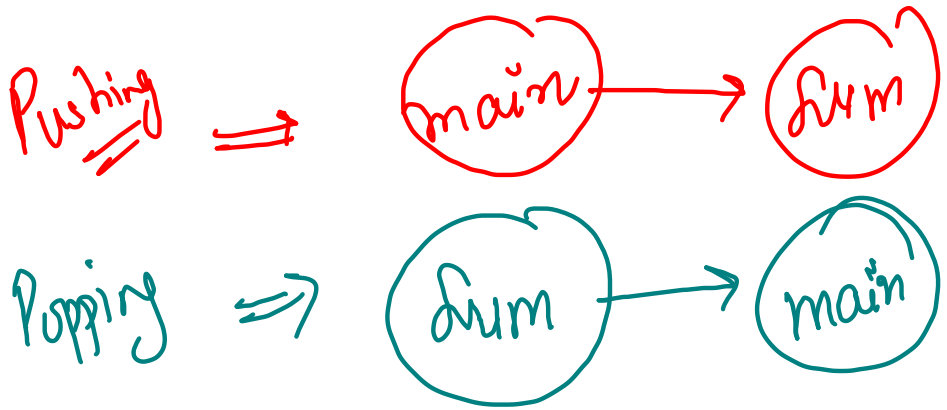
Functn call stack

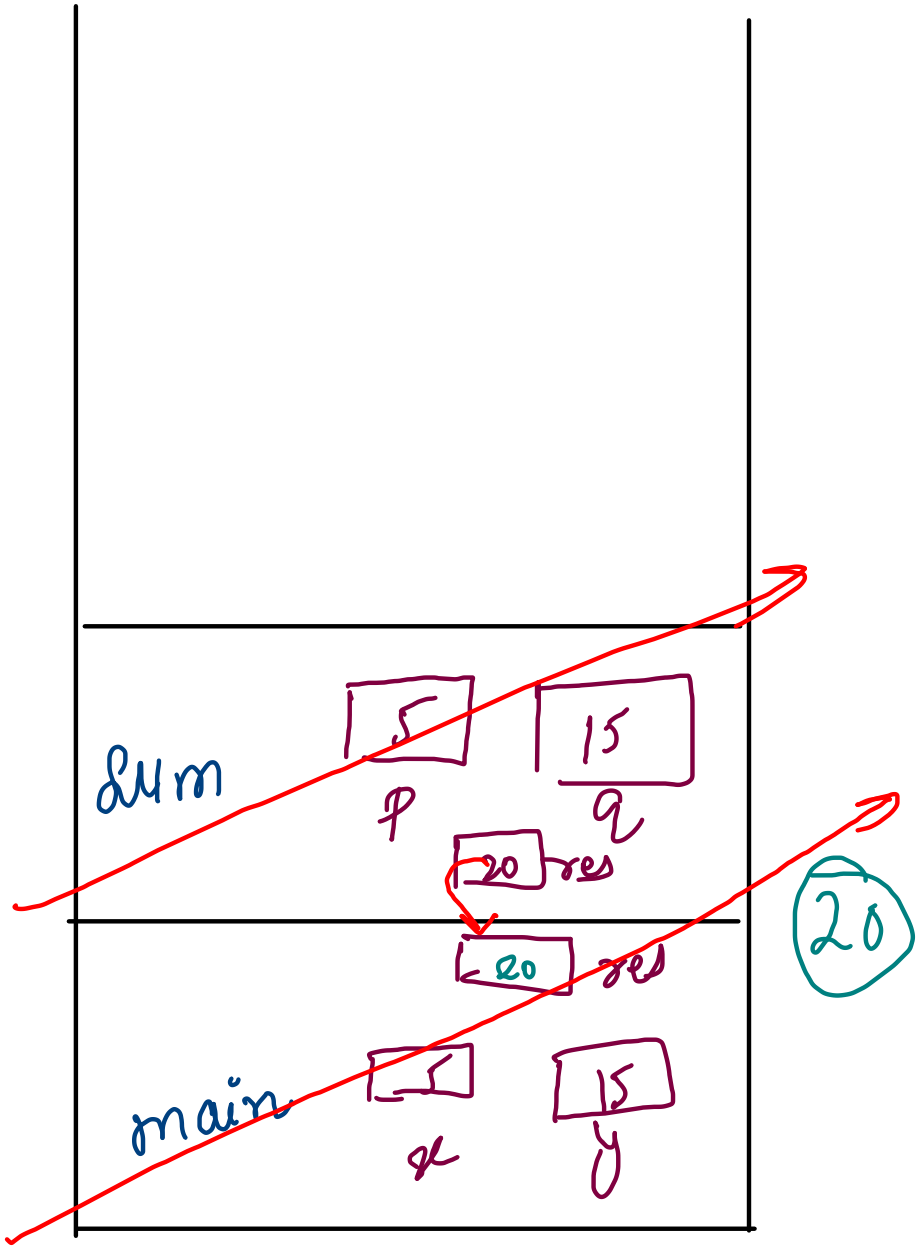
↳ functns will get pushed/popped

```java
public class Solution {
    public static void sum(int x, int y){
        System.out.println(x + y);
    }

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int x = scn.nextInt();
        int y = scn.nextInt();
        sum(x, y);
    }
}
```



```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int x = scn.nextInt();
    int y = scn.nextInt();
    int res = sum(x, y); // function call (WHEN)
    System.out.println(res);
}

// function body (WHAT)
public static int sum(int p, int q){
    int res = p + q;
    return res;
}
```



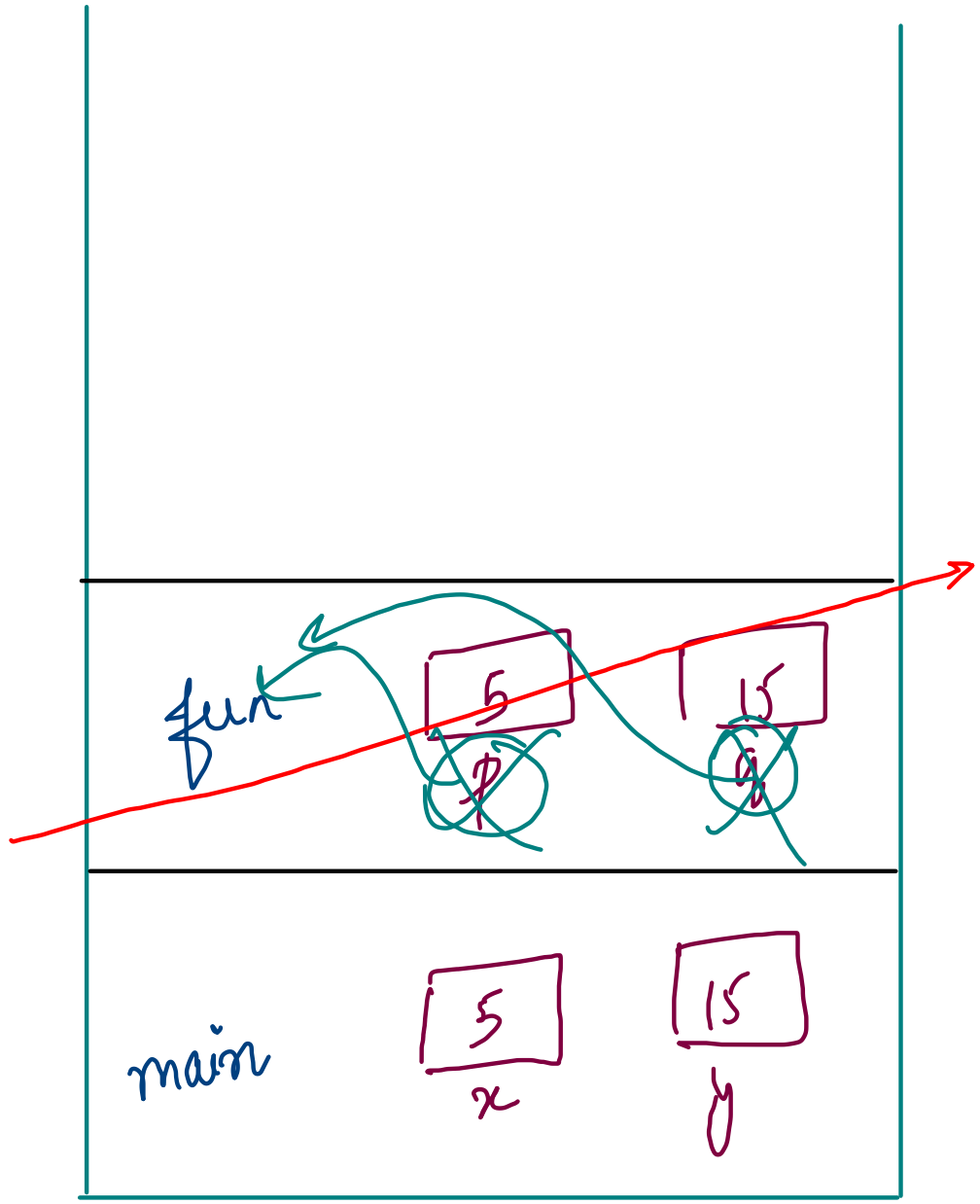


```java
public class Solution {
    public static void fun(int p, int q){
        System.out.println(p + " " + q);
    }

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int x = scn.nextInt();
        int y = scn.nextInt();

        fun(x, y);
        System.out.println(p + " " + q);
    }

}
```

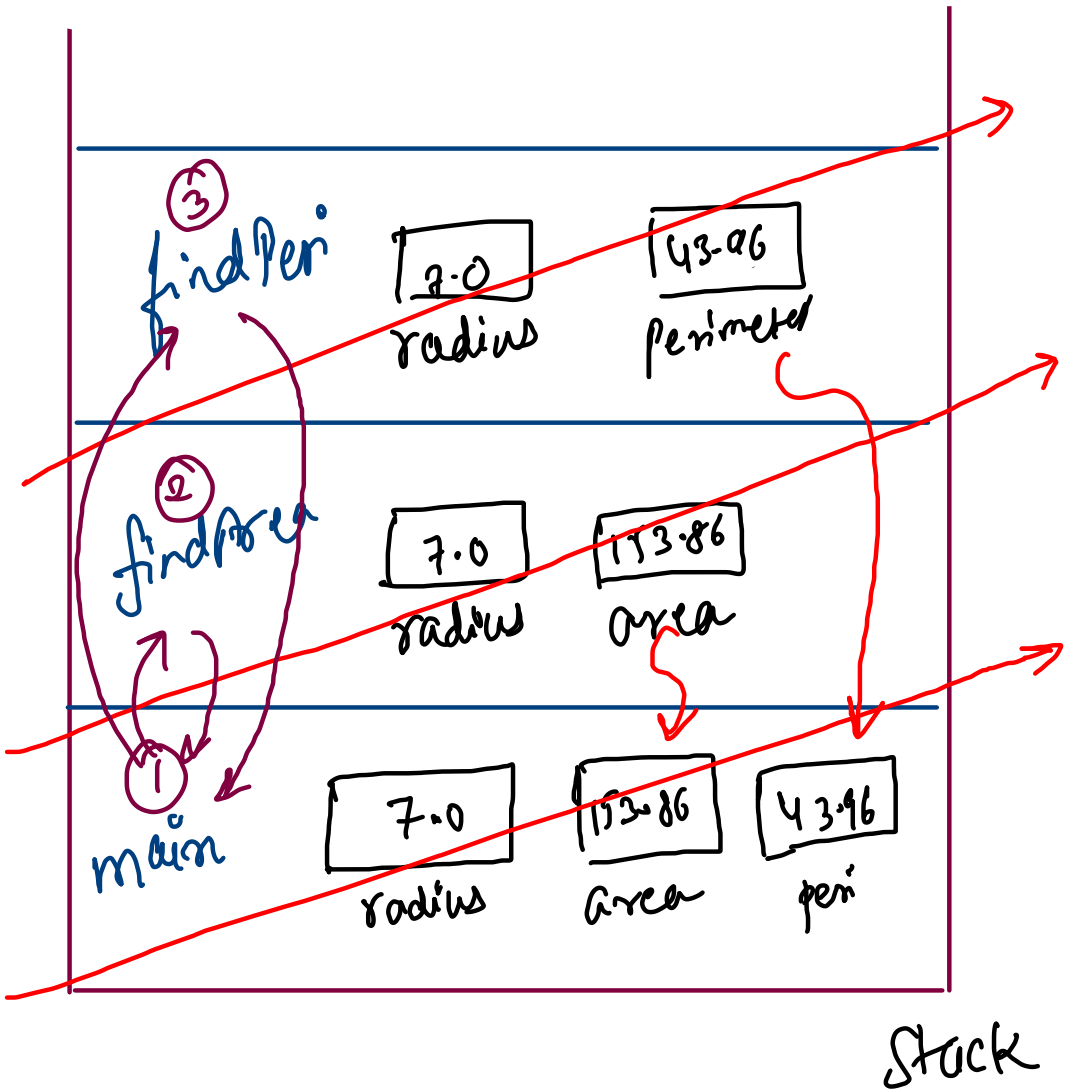
↳ Compilation Error

✓ 5 _ 15

5 - 15



```java
public class Solution {
    public static double findArea(double radius){
        double area = 3.14 * radius * radius;
        return area;
    }

    public static double findPerimeter(double radius){
        double perimeter = 3.14 * 2 * radius;
        return perimeter;
    }

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        double radius = scn.nextDouble();
        double area = findArea(radius);
        double perimeter = findPerimeter(radius);

        System.out.println(area);
        System.out.println(perimeter);
    }
}
```



① 153.86
② 43.96

Swap 2 variables

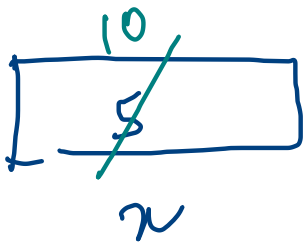
```
int x = 5;
int y = 10;
```

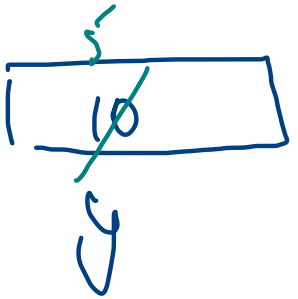
```
x = y;
y = x;
```

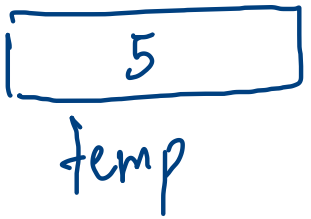
✗

```
int temp = x;
x = y;
y = temp;
```

✓





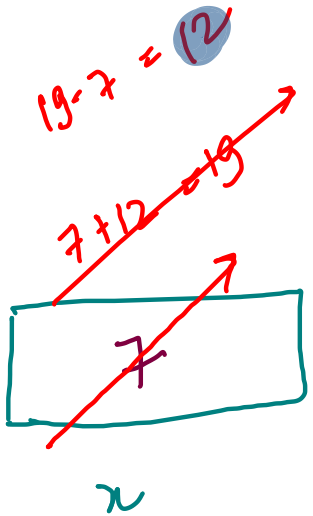


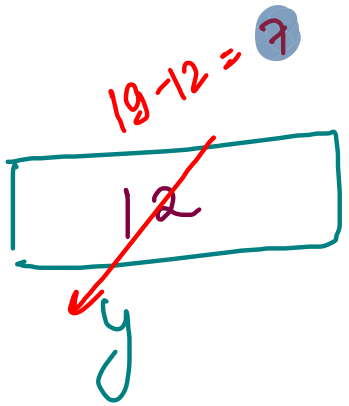
```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int x = scn.nextInt();
    int y = scn.nextInt();

    int c = x;

    System.out.println("c = " + c);

    x = y;
    y = c;

    System.out.println("x = " + x);
    System.out.println("y = " + y);

    System.out.println("x = " + x);
    System.out.println("y = " + y);

}
```

Swapping using third variable

1. int temp = x;
2. x = y;
3. y = temp;

Swapping without extra variable





$$x = x + y$$

$$y = x - y$$

$$x = x - y$$

```java
public class Solution {
    public static void swap(int x, int y){
        System.out.println(x + " " + y);

        int temp = x;
        x = y;
        y = temp;

        System.out.println(x + " " + y);
    }

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int x = scn.nextInt();
        int y = scn.nextInt();


        System.out.println(x + " " + y);

        swap(x, y);

        System.out.println(x + " " + y);
    }
}
```

A

7 12

7 12

12 7

12 7

B

7 12

7 12

7 12

12 7

C

7 12

7 12

12 7

7 12

D

7 12

7 12

7 12

7 12

Swap Answer

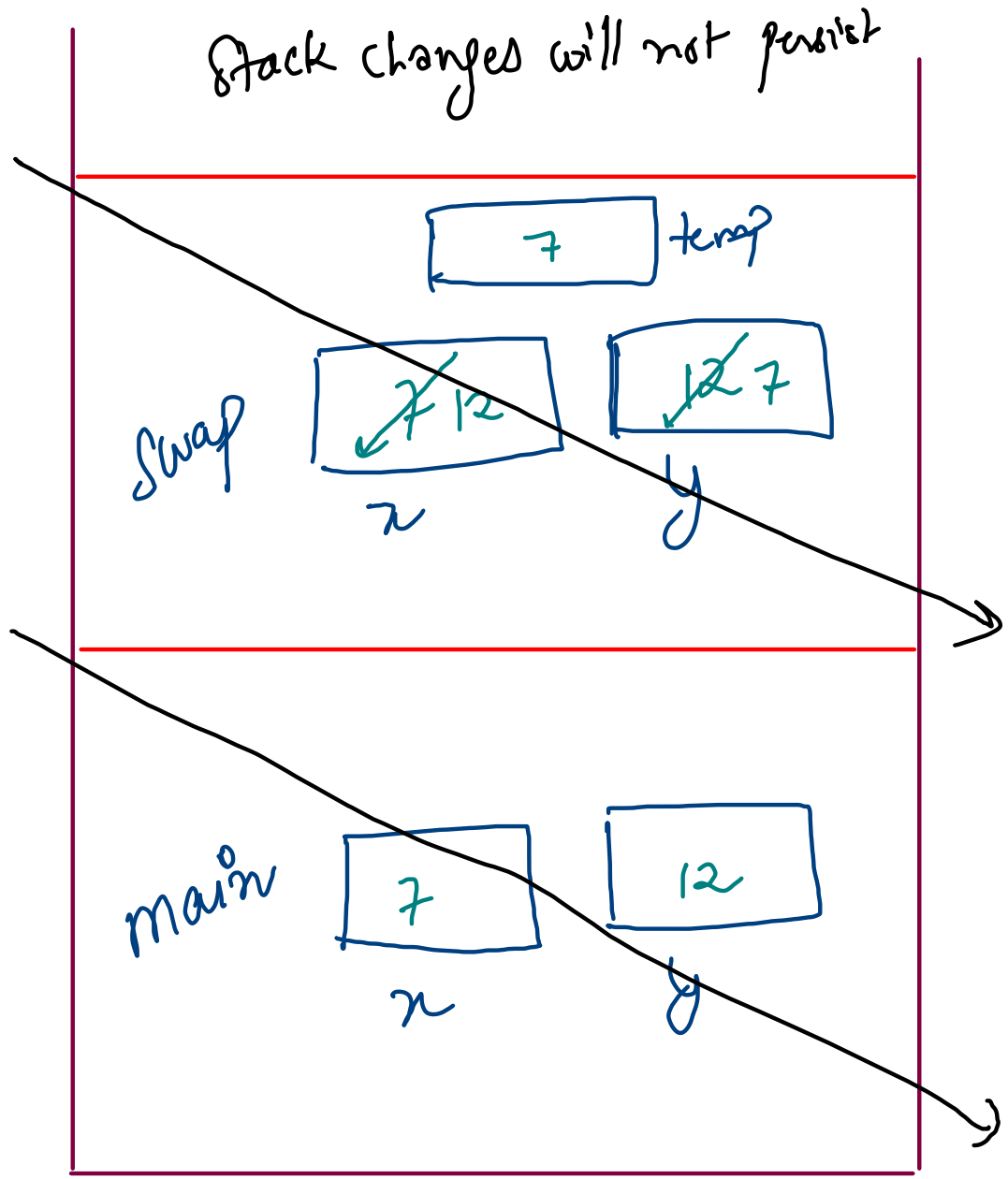
00:04:22 | 1 question | 90 of 143 (62%) participat...

1. Correct Output (Single Choice) *

90/90 (100%) answered

Choice 1	(30/90) 33%
Choice 2	(21/90) 23%
Choice 3	(37/90) 41%
Choice 4	(2/90) 2%

```java
public class Solution {
    public static void swap(int x, int y){
        System.out.println(x + " " + y); 7 12

        int temp = x;
        x = y;
        y = temp;

        System.out.println(x + " " + y); 12 7
    }

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int x = scn.nextInt(); 7
        int y = scn.nextInt(); 12


        System.out.println(x + " " + y); 7 12

        swap(x, y);

        System.out.println(x + " " + y); 7 12
    }
}
```



(3) ways to swap using swap()

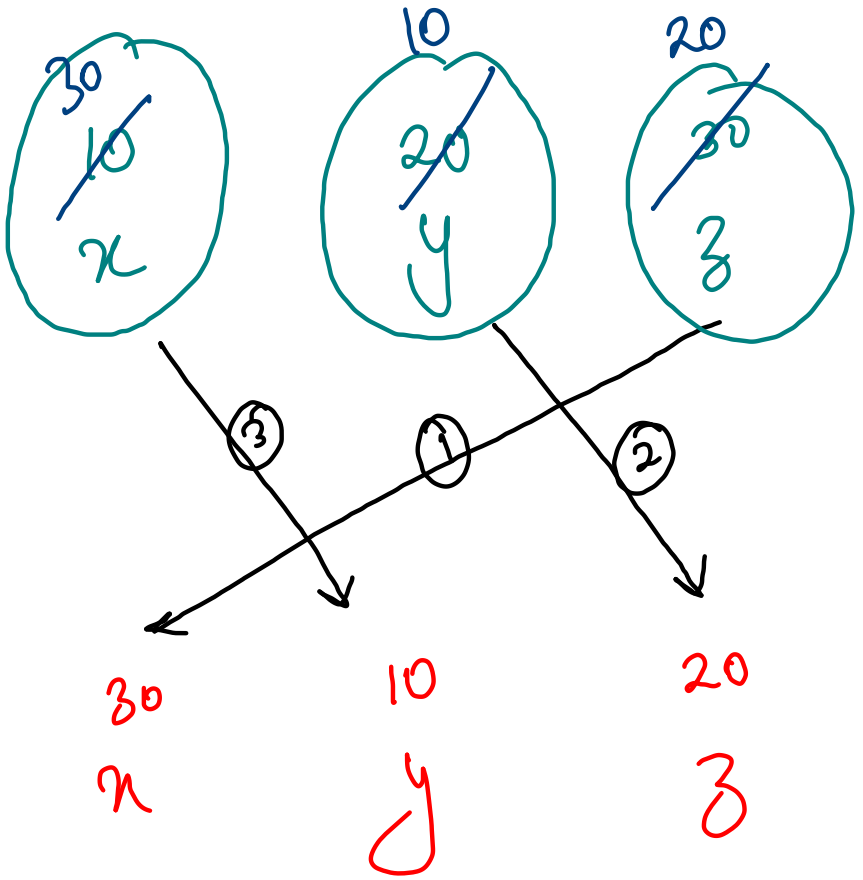
(1) Returning both the variables → Arrays

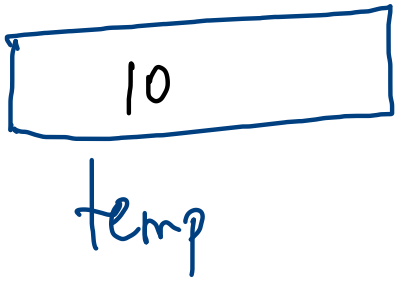
(2) Passing them in an Array

(3) Returning Object

} Heap memory

Swap x, y, z





① int temp = x

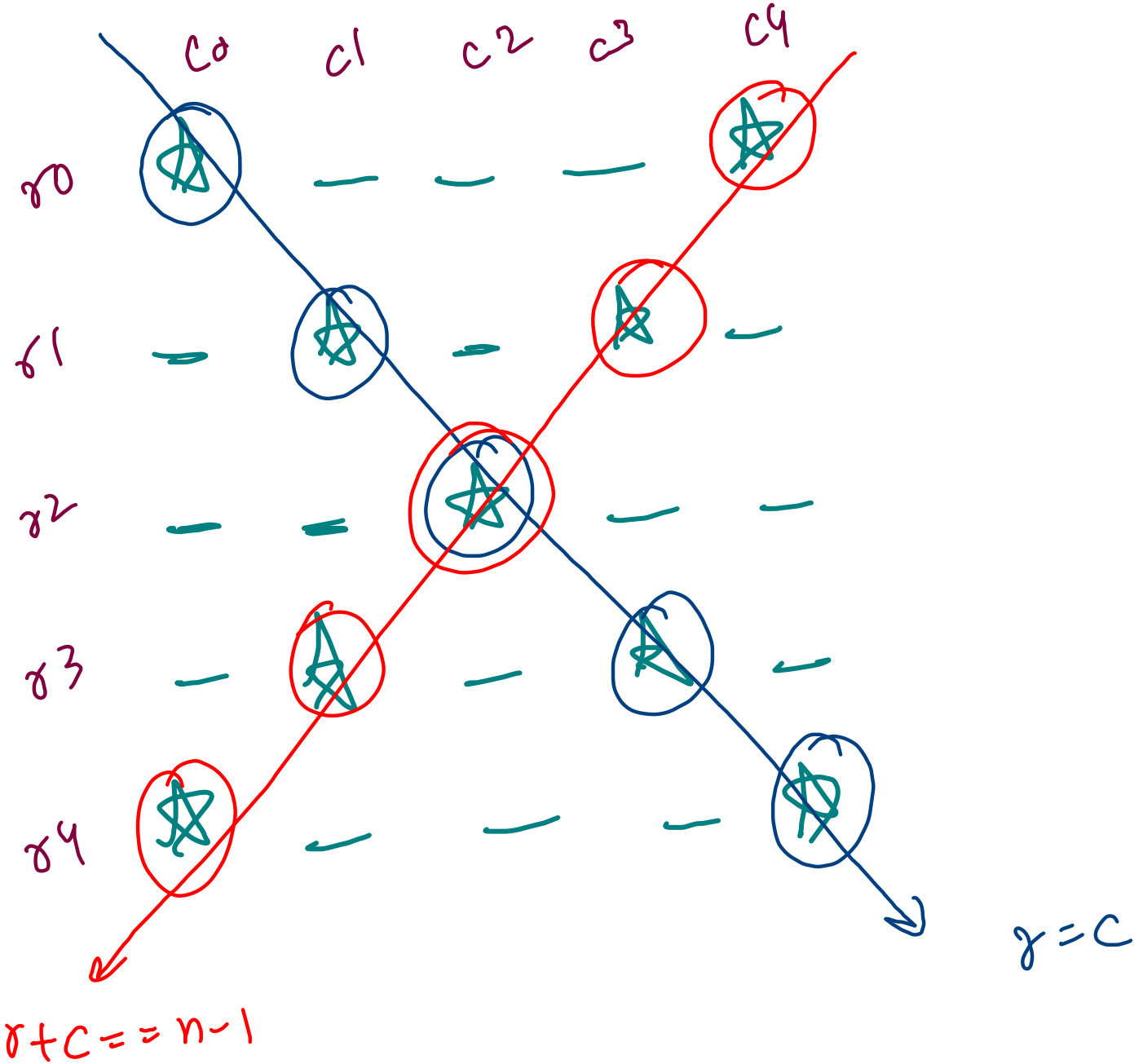
② x = z

③ z = y

④ y = temp

```java
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int x = scn.nextInt();
        int y = scn.nextInt();
        int z = scn.nextInt();

        int temp = x;
        x = z;
        z = y;
        y = temp;

        System.out.print(x + "\n" + y + "\n" + z);
    }
}
```

```java
import java.io.*;
import java.util.*;

public class Solution {
    public static int join(int x, int y){
        return 10 * x + y;
    }

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int x = scn.nextInt();
        int y = scn.nextInt();
        int res = join(x, y);
        System.out.println(res);
    }
}
```



0,4 = 4

1,3 = 4

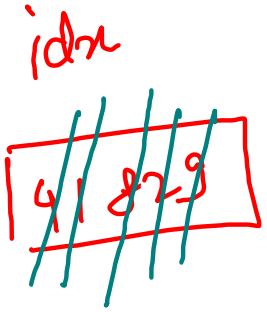
2,2 = 4

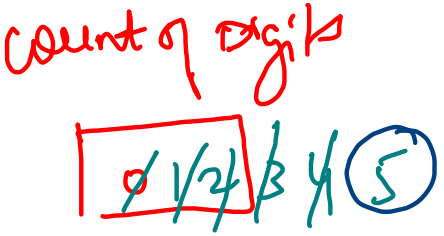
3,1 = 4

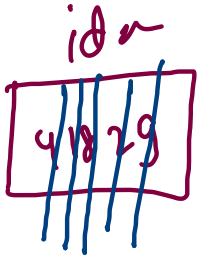
4,0 = 4

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();

    int countOfDigits = 0;
    for(int idx = n; idx > 0; idx = idx / 10){
        countOfDigits++;
    }

    System.out.println(countOfDigits);

    for(int idx = n; idx > 0; idx = idx / 10){
        System.out.println(idx % 10);
    }
}
```

N = 41829

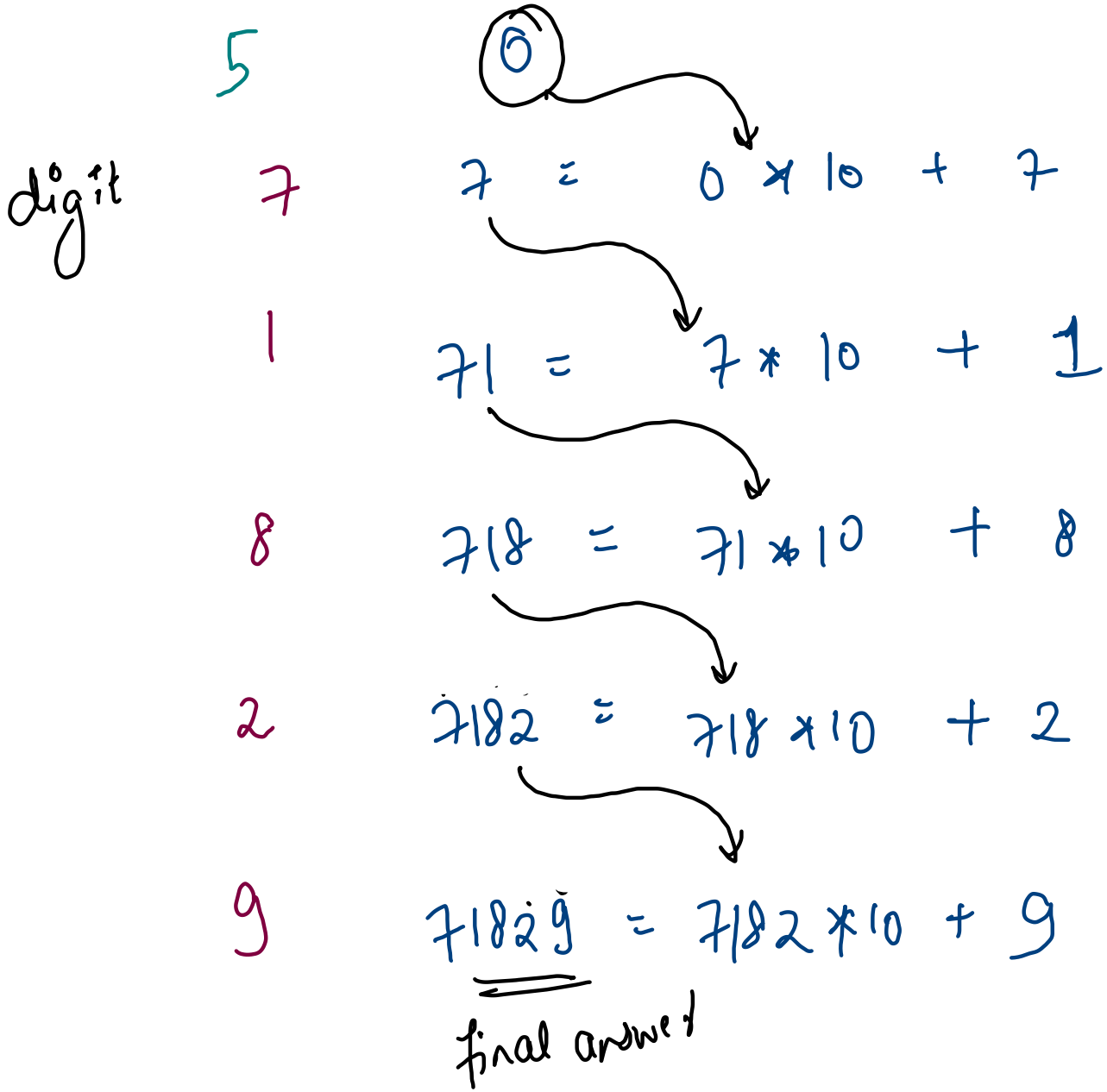






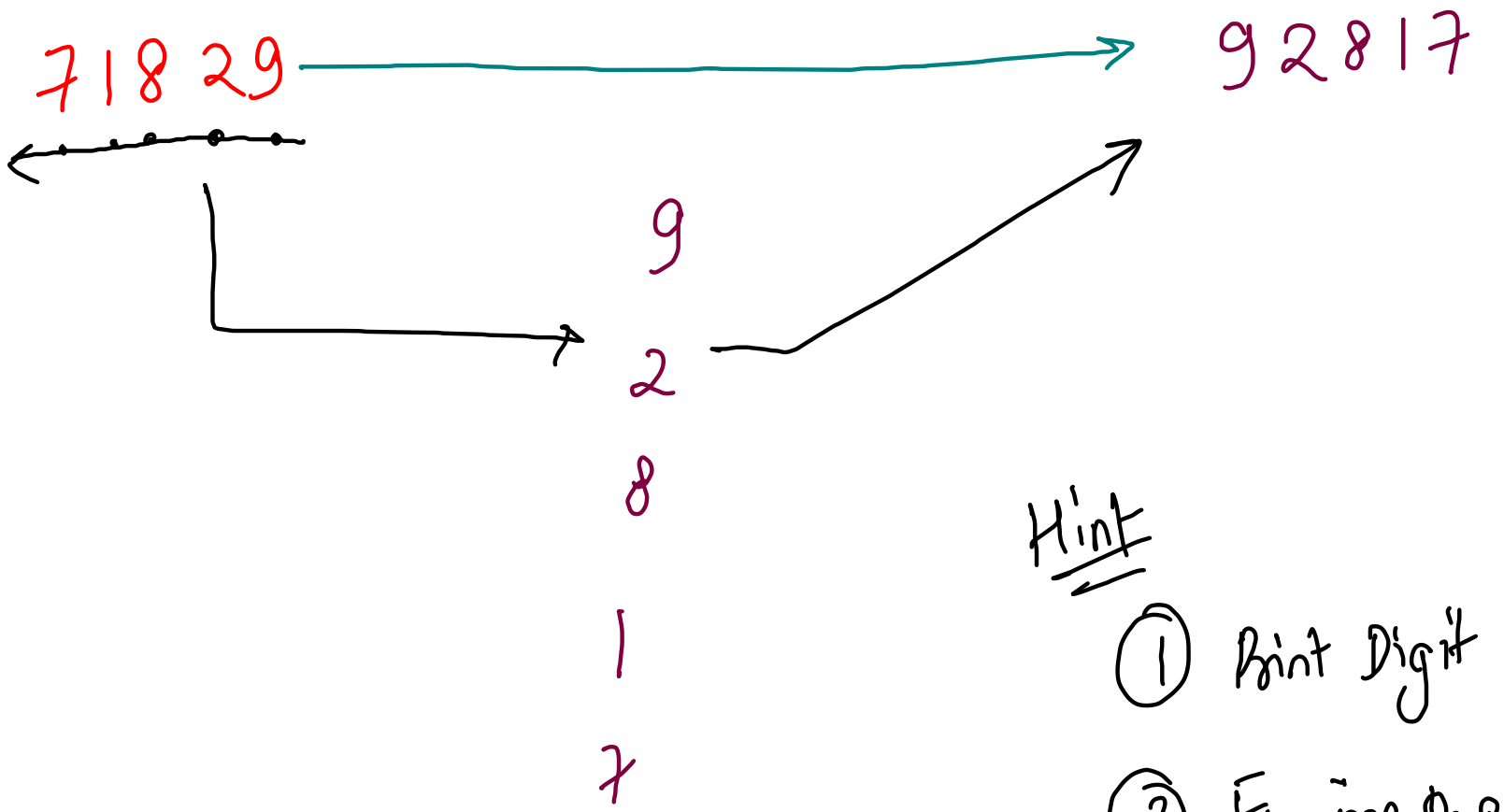
9, 2, 8, 1, 4

```java
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();

        for(int row=0; row<n; row++){
            for(int col=0; col<n; col++){
                if(row == col || row + col == n - 1){
                    System.out.print("*");
                } else {
                    System.out.print(" ");
                }
            }
            System.out.println();
        }
    }
}
```




```java
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int countOfDigits = scn.nextInt();
        int res = 0;

        for(int idx=0; idx<countOfDigits; idx++){
            int digit = scn.nextInt();
            res = res * 10 + digit;
        }

        System.out.println(res);
    }
}
```

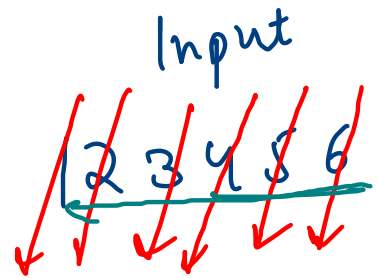
Reverse No



Hint

1. Print Digit by Digit
2. Forming the no from digits

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();

    int res = 0;
    for(int idx=n; idx>0; idx=idx/10){
        int digit = idx % 10;
        res = res * 10 + digit;
    }

    System.out.println(res);
}
```



res dig

0

6 = 0 * 10 + 6

65 = 6 * 10 + 5

654 = 65 * 10 + 4

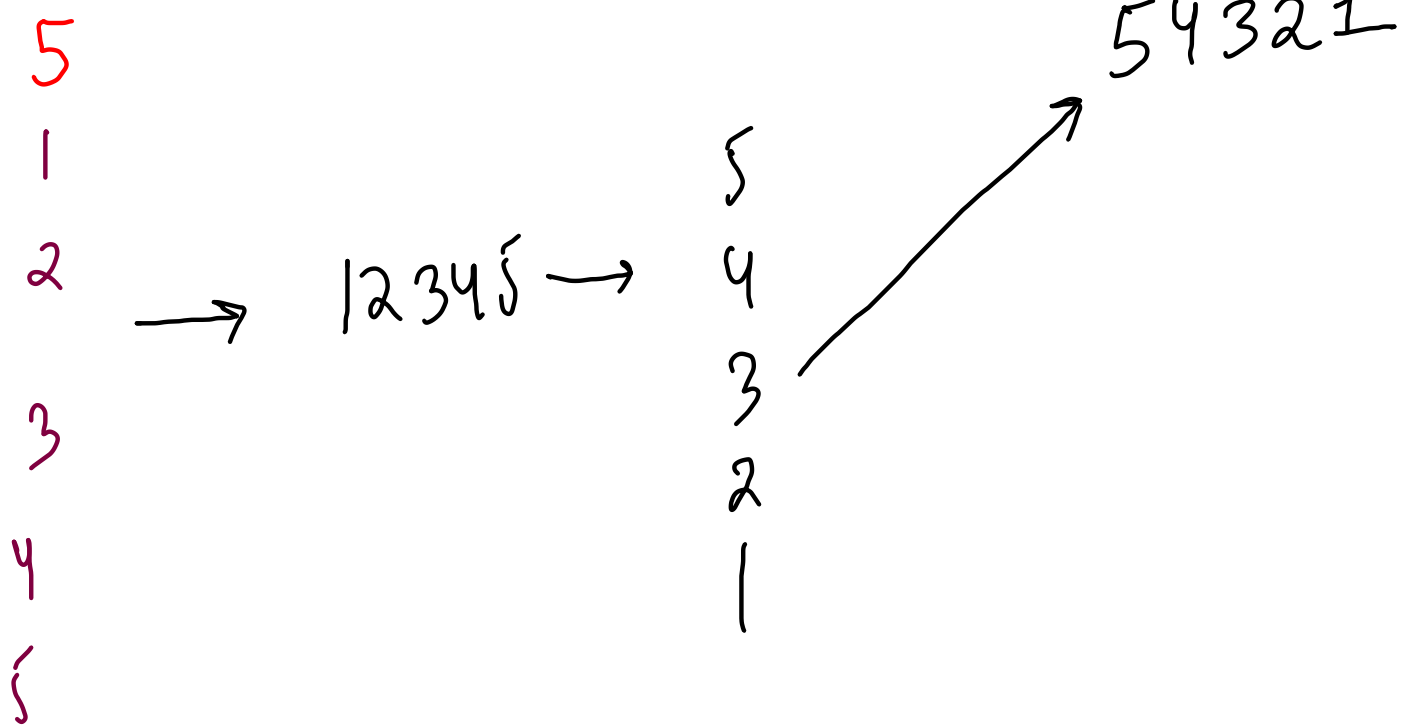
6543 = 654 * 10 + 3

65432 = 6543 * 10 + 2

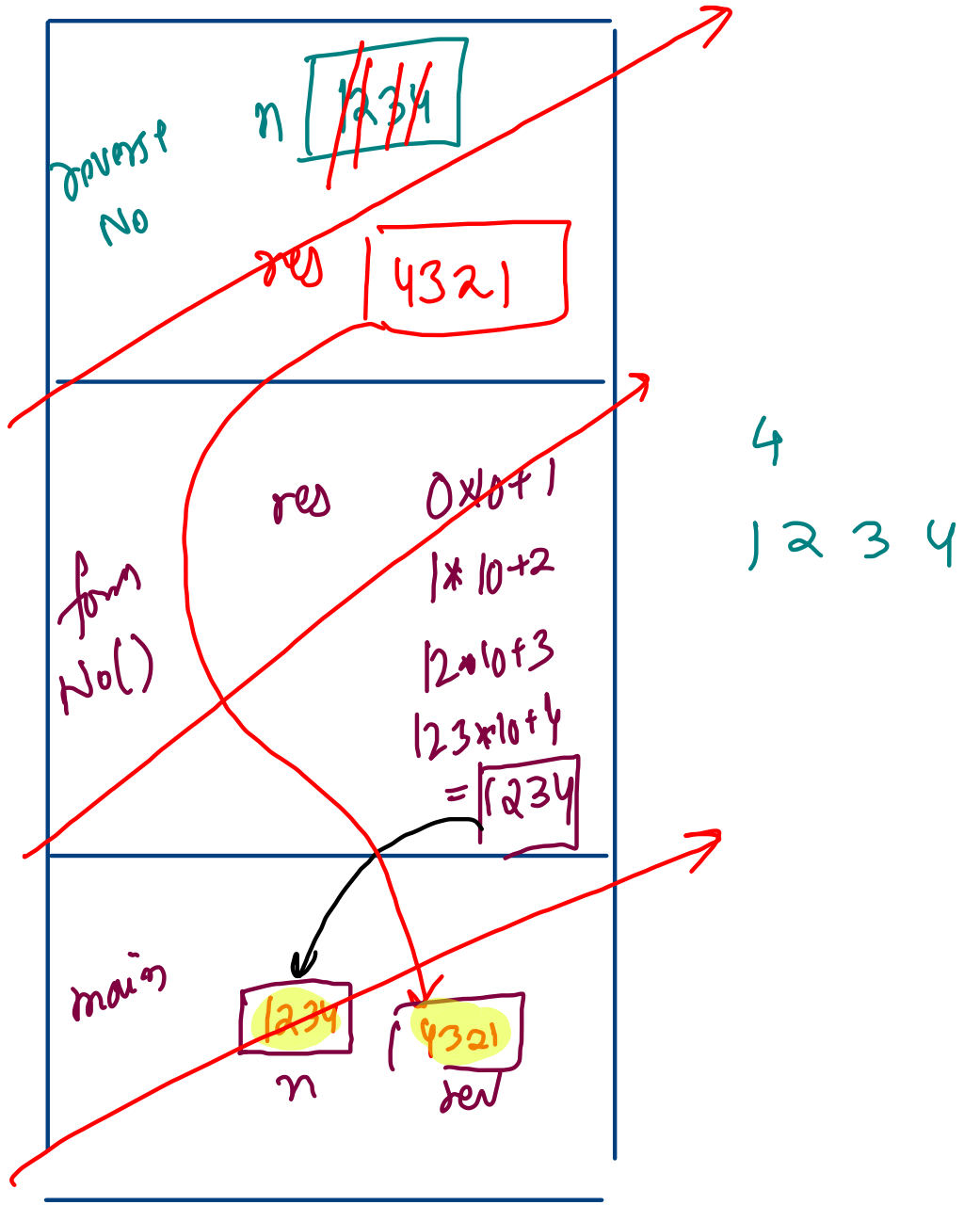
654321 = 65432 * 10 + 1

↳ res

Integer (Reverse)



```java
public static int reverseNumber(int n){
    int res = 0;
    for(int idx=n; idx>0; idx=idx/10){
        int digit = idx % 10;
        res = res * 10 + digit;
    }

    return res;
}

public static int formNumber(){
    Scanner scn = new Scanner(System.in);
    int countOfDigits = scn.nextInt();
    int res = 0;

    for(int idx=0; idx<countOfDigits; idx++){
        int digit = scn.nextInt();
        res = res * 10 + digit;
    }

    return res;
}

public static void main(String[] args) {
   int n = formNumber();
   int rev = reverseNumber(n);
   System.out.println(n + "\n" + rev);
}
```



4

1 2 3 4

Armstrong Number

$153 = 1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153$ true

$370 = 3^3 + 7^3 + 0^3 = 27 + 343 + 0 = 370$ true

$111 = 1^3 + 1^3 + 1^3 = 3$ false

1. digits
2. cube of digits
3. sum of cube of digits == No

$1 = 1^3 = 1$ true

$0 = 0^3 = 0$ true

✓ ✓ ✓

153

0+

$3 \rightarrow 3^3 = 27$

+

$5 \rightarrow 5^3 = 125$

+27

$1 \rightarrow 1^3 = 125 + 27 + 1$

$= 153$

- Additive Property

$a + 0 = a$

$0 + a = a$

- multiplicative Property

$1 * a = a * 1 = a$

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int res = 0;

    for(int idx = n; idx > 0; idx = idx / 10){
        int digit = idx % 10; // Digit
        int cube = digit * digit * digit; // Cube
        res = res + cube; // Sum
    }

    if(res == n) System.out.println(true);
    else System.out.println(false);
}
```

```java
public class Solution {
    public static boolean isAmstrongNo(int n){
        int res = 0;

        for(int idx = n; idx > 0; idx = idx / 10){
            int digit = idx % 10; // Digit
            int cube = digit * digit * digit; // Cube
            res = res + cube; // Sum
        }

        if(res == n) return true;
        else return false;
    }
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        System.out.println(isAmstrongNo(n));
    }
}
```

$n = 1634$

idx = 1634

$res = 0 + 4^3 + 3^3 + 6^3 + 1^3$

$= 64 + 27 + 216 + 1$

$\neq 1634$

false

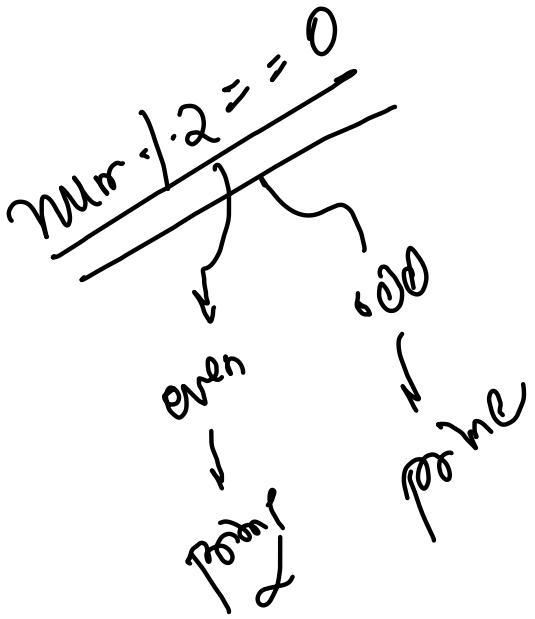
```java
public static boolean isAmstrongNo(int n){
    int res = 0;

    for(int idx = n; idx > 0; idx = idx / 10){
        int digit = idx % 10; // Digit
        int cube = digit * digit * digit; // Cube
        res = res + cube; // Sum
    }

    if(res == n) return true;
    else return false;
}

public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int left = scn.nextInt();
    int right = scn.nextInt();

    for(int idx=left; idx<=right; idx++){
        if(isAmstrongNo(idx) == true){
            System.out.println(idx);
        }
    }
}
```

Print Amstrong Nos in a given Range

Prime No's

↳ only having 2 factors → 1
→ number itself

Eg 1, 2, 3, 5, 7, 11, 13, 17, 19, 23



Approach ① Check Divisibility from [2, n-1]

49

↓

Composite

[2, 48]

49 % 2 ≠ 0

49 % 3 ≠ 0

49 % 4 ≠ 0

49 % 5 ≠ 0

49 % 6 ≠ 0

49 % 7 == 0

↳ composite

17 % 2 ≠ 0

17 % 3 ≠ 0

17 % 4 ≠ 0

17 % 5 ≠ 0

17 % 6 ≠ 0

17 % 7 ≠ 0

17 % 8 ≠ 0

17 % 9 ≠ 0

17 % 10 ≠ 0

17

↓

prime

17 % 11 ≠ 0

17 % 12 ≠ 0

17 % 13 ≠ 0 [2, 16]

17 % 14 ≠ 0

17 % 15 ≠ 0

17 % 16 ≠ 0

17 ⇒ prime

```java
public static boolean isPrime1(int n){
    if(n == 1 || n == 2) return true; // 1 and 2 are prime

    for(int idx = 2; idx <= n - 1; idx++){
        // is idx a factor of n OR is n a multiple of idx
        if(n % idx == 0) return false;
    }

    return true;
}

public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    boolean res = isPrime1(n);

    if(res == true){
        System.out.println("Yes");
    } else {
        System.out.println("No");
    }
}
```

→ n has factor (idx) ∴ not prime

→ no factor of n other than {1, n} ∴ n is prime

Prime

13

idx = 2
13%2 ≠ 0
13%3 ≠ 0
13%4 ≠ 0
13%5 ≠ 0
13%6 ≠ 0
13%7 ≠ 0
13%8 ≠ 0
13%9 ≠ 0
13%10 ≠ 0
13%11 ≠ 0
13%12 ≠ 0

11 times loop is running

yes, prime

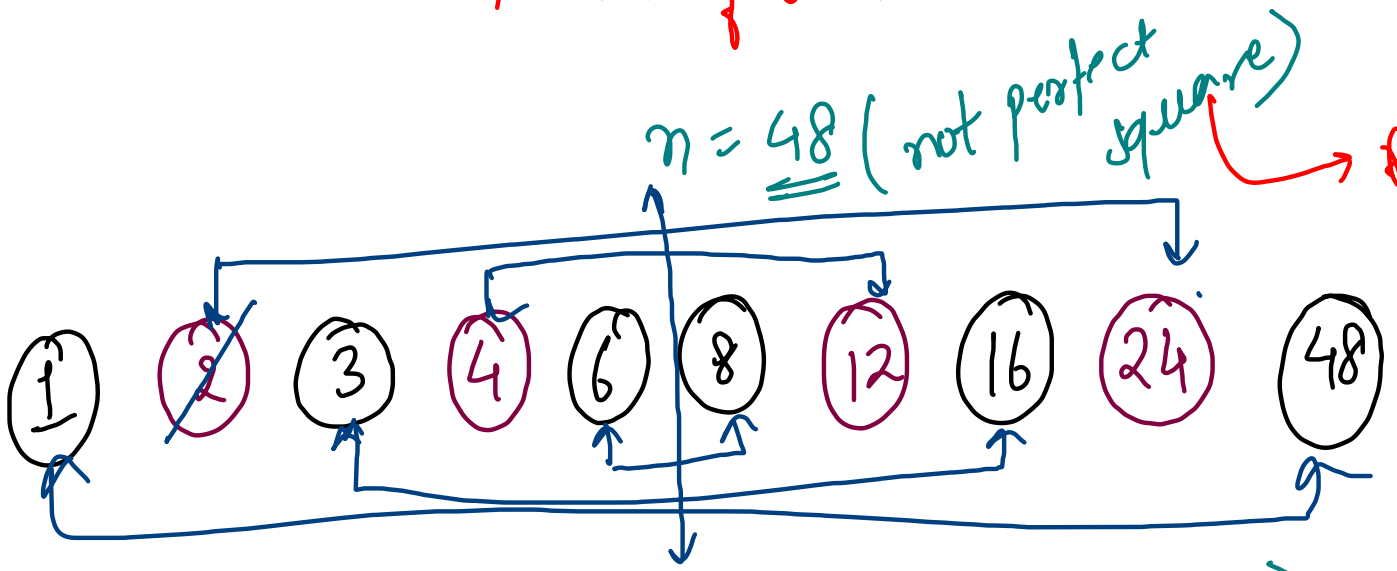
Composite

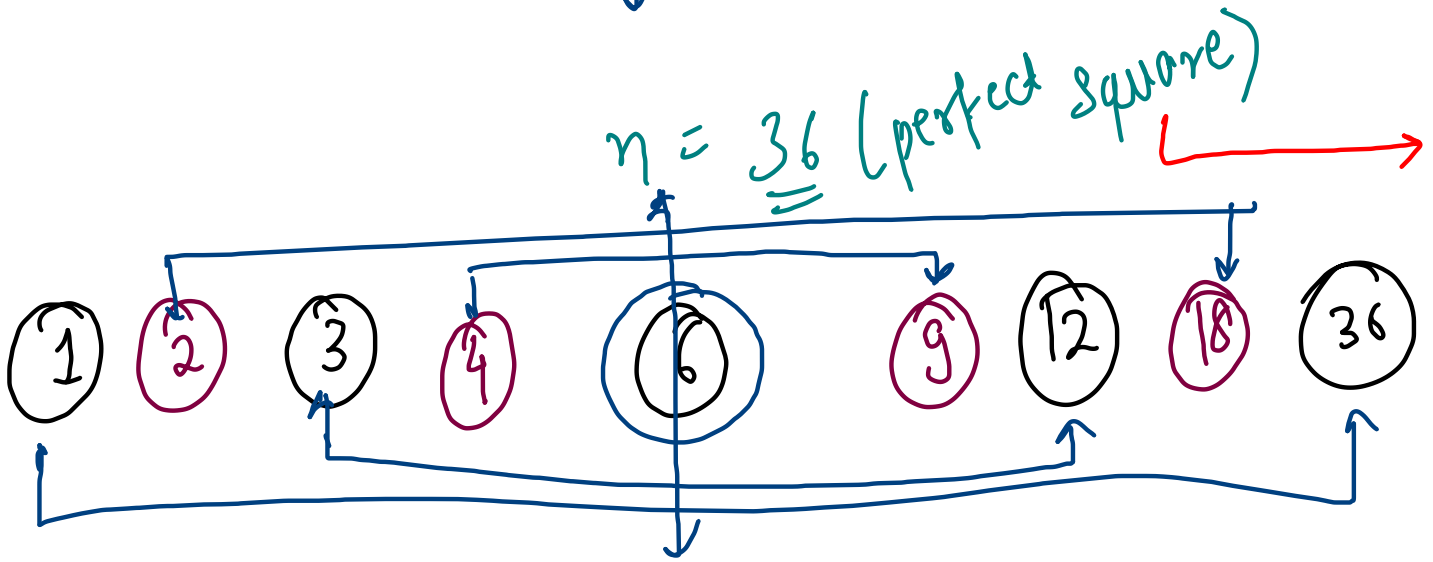
35

idx = 2
35%2 ≠ 0
idx = 3
35%3 ≠ 0
idx = 4
35%4 ≠ 0
idx = 5
35%5 = 0

return false

Factors of a No





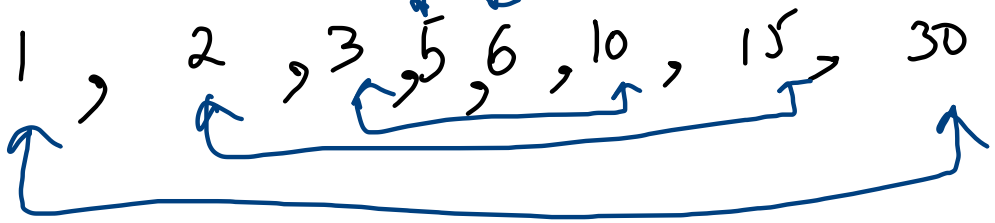
29

$\sqrt{29} = 6$

[2, 6]

no factor in b/w [2, $\sqrt{N}$]

30



```java
public static boolean isPrime2(int n){
    if(n == 1 || n == 2) return true; // 1 and 2 are prime

    int sqrt = (int)Math.sqrt(n);

    for(int idx = 2; idx <= sqrt; idx++){
        // is idx a factor of n OR is n a multiple of idx
        if(n % idx == 0) return false;
    }

    return true;
}

public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    boolean res = isPrime2(n);

    if(res == true){
        System.out.println("Yes");
    } else {
        System.out.println("No");
    }
}
```

$n = 49$

$[2, 3, 4, 5, 6, 7]$

↳ $\sqrt{N}$ times

$n = 97$

prime

$[2, 3, 4, 5, 6, 7, 8, 9]$

$\sqrt{97}$

```java
public class Solution {
    public static boolean isPrime(int n){
        if(n == 1 || n == 2) return true; // 1 and 2 are prime

        int sqrt = (int)Math.sqrt(n);

        for(int idx = 2; idx <= sqrt; idx++){
            // is idx a factor of n OR is n a multiple of idx
            if(n % idx == 0) return false;
        }

        return true;
    }

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int size = scn.nextInt();

        for(int idx = 0; idx < size; idx++){
            int n = scn.nextInt();

            boolean res = isPrime(n);
            if(res == true){
                System.out.println("prime");
            } else {
                System.out.println("not prime");
            }
        }
    }
}
```

Checking each number is prime or not !

Better Approach :→

→ Sieve of eratosthenes

N = 48

1, 48

2, 24

3, 16

4, 12

6, 8

Factors of No

[1, 48]

48 % 1 = 0 (1)

48 % 2 = = 0 (2)

48 % 3 = = 0 (3)

48 % 4 = = 0 (4)

48 % 5 ≠ 0

48 % 6 = 0 (6)

48 % 7 ≠ 0

48 % 8 = 0 (8)

48 % 9 ≠ 0

48 % 10 ≠ 0

48 % 11 ≠ 0

48 % 12 = 0 (12)

⋮

48 % 48 = 0 (48)

```java
public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();

        for(int idx=1; idx<=n; idx++){
            if(n % idx == 0){
                System.out.println(idx);
            }
        }
    }
}
```

n = 12

idx → [1, 12]

idx = 1 ✓ 12%1==0

2 ✓ 12%2==0

3 ✓ 12%3==0

4 ✓ 12%4==0

5 12%5≠0

6 ✓ 12%6=0

7 12%7≠0

8 12%8≠0

9 12%9≠0

10 12%10≠0

11 12%11≠0

12 ✓ 12%12=0

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();

    for(int idx=1; idx <= n/2; idx++){
        if(n % idx == 0){
            System.out.println(idx);
        }
    }

    System.out.println(n);
}
```

Optimization

[1, n/2]

n = 12

idx → [1, 12]

idx = 1 ✓ 12%1==0

2 ✓ 12%2==0

3 ✓ 12%3==0

4 ✓ 12%4==0

5 12%5≠0

6 ✓ 12%6=0

12 ✓ 12%12=0

$n = 48$

1, 2, 3, 4, 6, 8, 12, 16, 24, 48

$n = 30$

1, 2, 3, 5, 6, 10, 15, 30

[2,5] ∝

[2,28] ∝ $n = 29$ ↳ $\sqrt{29}$ (5)

```java
public static boolean isPrime(int n){
    int sqrt = (int)(Math.sqrt(n));

    for(int idx=2; idx<=sqrt; idx++){
        if(n % idx == 0) return false;
    }

    return true;
}

public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    for(int idx=2; idx<=n; idx++){
        if(n % idx == 0 && isPrime(idx) == true){
            System.out.println(idx);
        }
    }
}
```

eg1 $n = 2*2*2*3*3*5*5*7 = 63000$

steps = $0+2+2+2+3+3+5+5+5 =$

---

eg2 $n = 100 = 2*2*5*5$

steps = $20+2+2+5+5 = 34$

$100/2 = 50/2 = 25/2\alpha$

$25/3\alpha$

$25/5 = 5/5 = 1$

---

eg3 $n = 210$

steps = $7+2+3+5 = 17$

$210/2 = 105/2\alpha$

$105/3 = 35/3\alpha$

$35/5 = 7/5\alpha$

```
while ( n%2==0 || n%3==0 || n%5==0 ) {

    if (n%2==0) {
        n = n/2;        steps = steps+2;
    } else if (n%3==0) {
        n = n/3;        steps = steps+3;
    } else {
        n = n/5;        steps = steps+5;
    }
}

Sysо( steps + "\n" + n);
```

n =	steps = 100
2100	
↓ /2	+2 ↓
1050	102
↓ /2	+2 ↓
525	104
↓ /3	+3 ↓
175	107
↓ /5	+5 ↓
35	112
↓ /5	+5 ↓
7	117

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int steps = scn.nextInt();

    while(n % 2 == 0 || n % 3 == 0 || n % 5 == 0){
        if(n % 2 == 0){
            n = n / 2;
            steps = steps + 2;
        } else if(n % 3 == 0){
            n = n / 3;
            steps = steps + 3;
        } else {
            n = n / 5;
            steps = steps + 5;
        }
    }

    System.out.println(steps + "\n" + n);
}
```

Print Character at 3rd index

```
String str = scn.nextLine();
```

"archit" ⇒ (6) strings length ⇒ str.length() = n [0, n-1]

0 1 2 3 4 5

```
char ch0 = str.charAt(0) ⇒ 'a'
char ch1 = str.charAt(1) ⇒ 'r'
char ch2 = str.charAt(2) ⇒ 'c'
char ch3 = str.charAt(3) ⇒ 'h'
char ch4 = str.charAt(4) ⇒ 'i'
char ch5 = str.charAt(5) ⇒ 't'
```

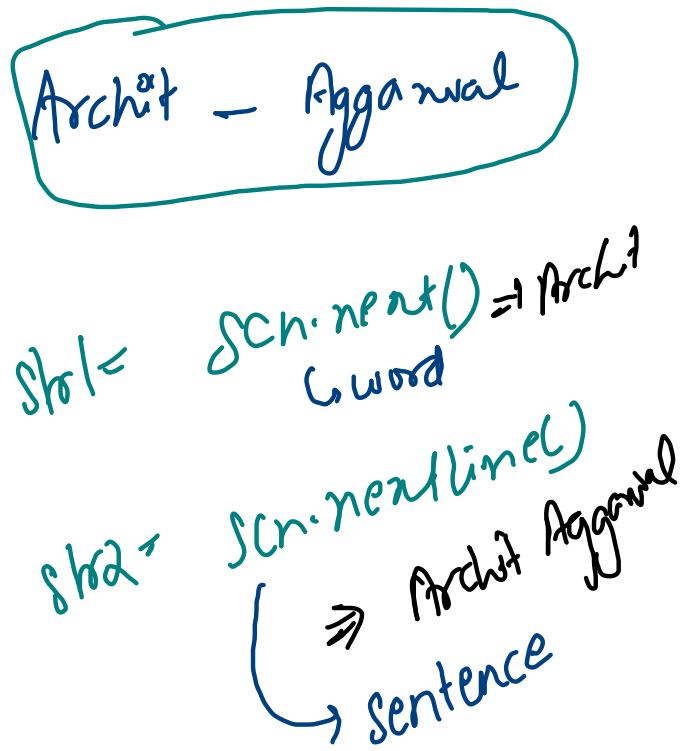
str.charAt(6) ⇒ index out of bound exception

str.charAt(-1) ⇒

```java
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        String str = scn.nextLine();

        if(str.length() >= 4){
            System.out.println(str.charAt(3));
        } else {
            System.out.println("Small string");
        }
    }
}
```

Concatenatn of (2) strings

```java
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        String str1 = scn.nextLine();
        String str2 = scn.nextLine();
        System.out.println(str1 + str2);
    }
}
```



String concatenate - ②

eg1 str1 = "hi" < str2 = "Hello"

hiHello hi

str1.length()
< str2.length()
str1 + str2 + str1

eg2 str1 = "namaste"

str2 = "ramram"

ramram namaste ramram

str2 + str1 + str2

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    String str1 = scn.nextLine();
    String str2 = scn.nextLine();

    if(str1.length() < str2.length()){
        System.out.println(str1 + str2 + str1);
    } else {
        System.out.println(str2 + str1 + str2);
    }
}
```

Print alternate characters

archit ⇒ aci

0 1 2 3 4 5

Saurabh ⇒ Suah

0 1 2 3 4 5 6

```
Syso(str);

for(int idx=0;idx<str.length();idx++)
{
    Syso(str.charAt(idx));
}
```

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    String str = scn.nextLine();

    for(int idx = 0; idx < str.length(); idx = idx + 2){
        System.out.print(str.charAt(idx));
    }
}
```

Directly Printing

Using Return

```java
public class Solution {
    public static String printAlternate(String str){
        String res = "";
        for(int idx = 0; idx < str.length(); idx = idx + 2){
            res = res + str.charAt(idx);
        }
        return res;
    }

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        String str = scn.nextLine();
        String res = printAlternate(str);
        System.out.println(res);
    }
}
```

Str = archit

0 1 2 3 4 5

res = "" + 'a'
= "a" + 'c'
= "ac" + 'i'
= "aci"

Reverse String

```
String res = "";
for(int idx = str.length()-1; idx >=0 ; idx--)
    res = res + str.charAt(idx);
```

str = "0 1 2 3 4 5"
"archit"

res = "" + 't'

= "t" + 'i' = "ti"

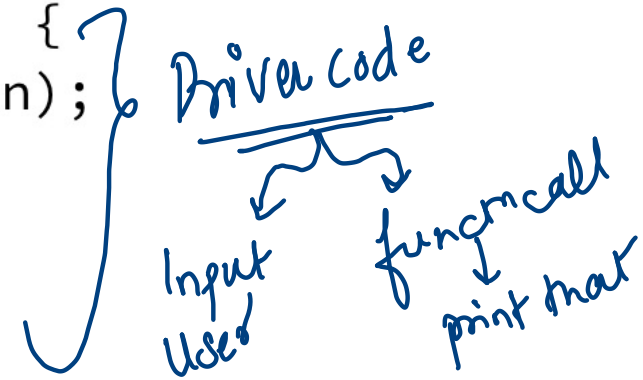
= "ti" + 'h' = "tih"

= "tih" + 'c' = "tihc"

= "tihc" + 'r' = "tihcr"

= "tihcr" + 'a' = "tihcra"

```java
public class Solution {
    public static String reverseString(String str){
        String res = "";
        for(int idx = str.length() - 1; idx >= 0; idx--){
            res = res + str.charAt(idx);
        }
        return res;
    }

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        String str = scn.nextLine();
        String res = reverseString(str);
        System.out.println(res);
    }
}
```



GCD (Greatest Common Divisor)

$\Leftrightarrow$

HCF (Highest Common Factor)

Approach 1 — Prime Factorization

2	36, 48
2	18, 24
3	9, 12
	3, 4

hcf or gcd $= 2 * 2 * 3 = 12$

Approach 2 — Long Division

$36 \overline{)48} \; 1$

-36

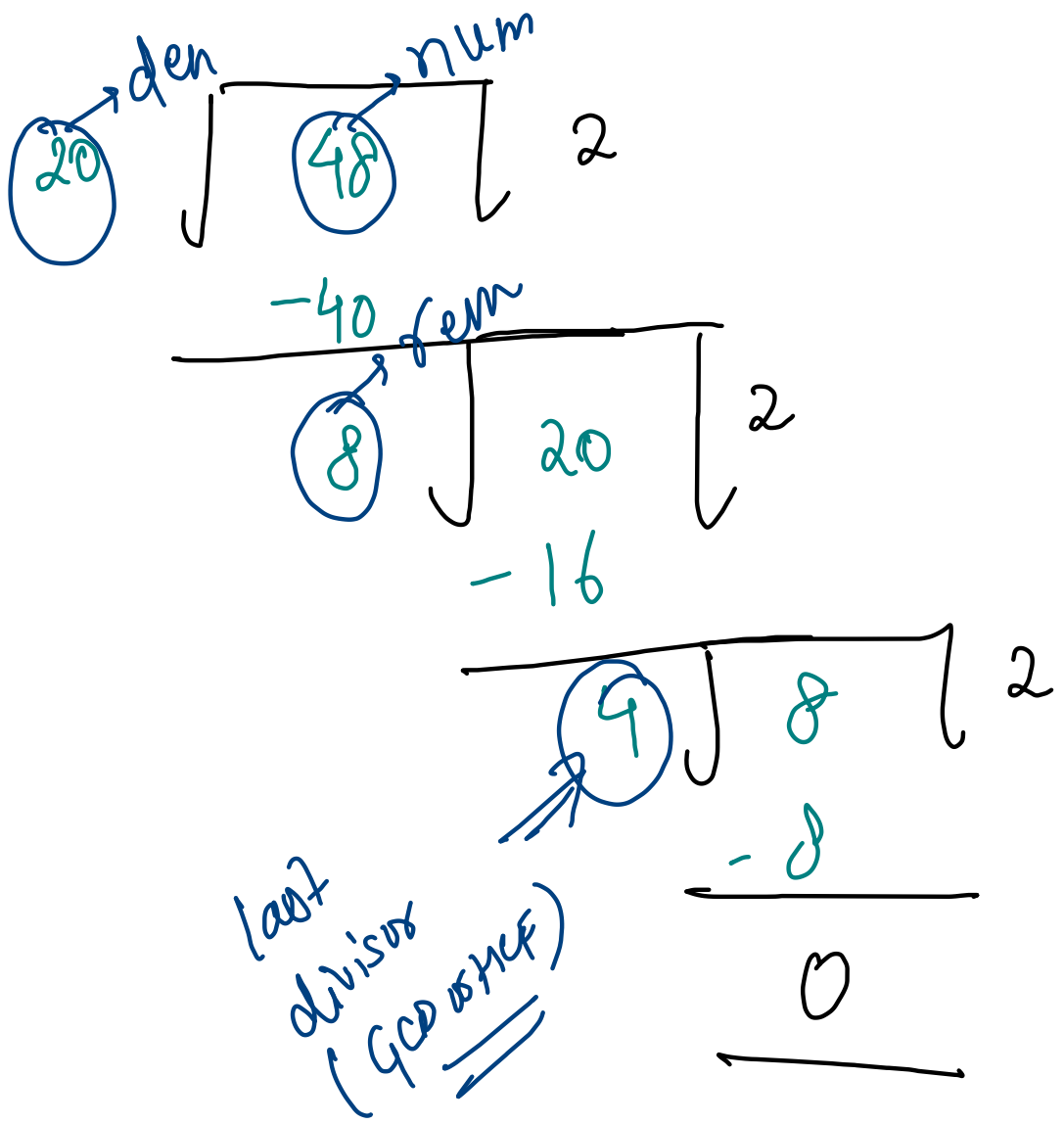
$12 \overline{)36} \; 3$

-36

0

last divisor ↓ ans gcd

GCD (20, 48)



$20 = 2 * 2 * 5$

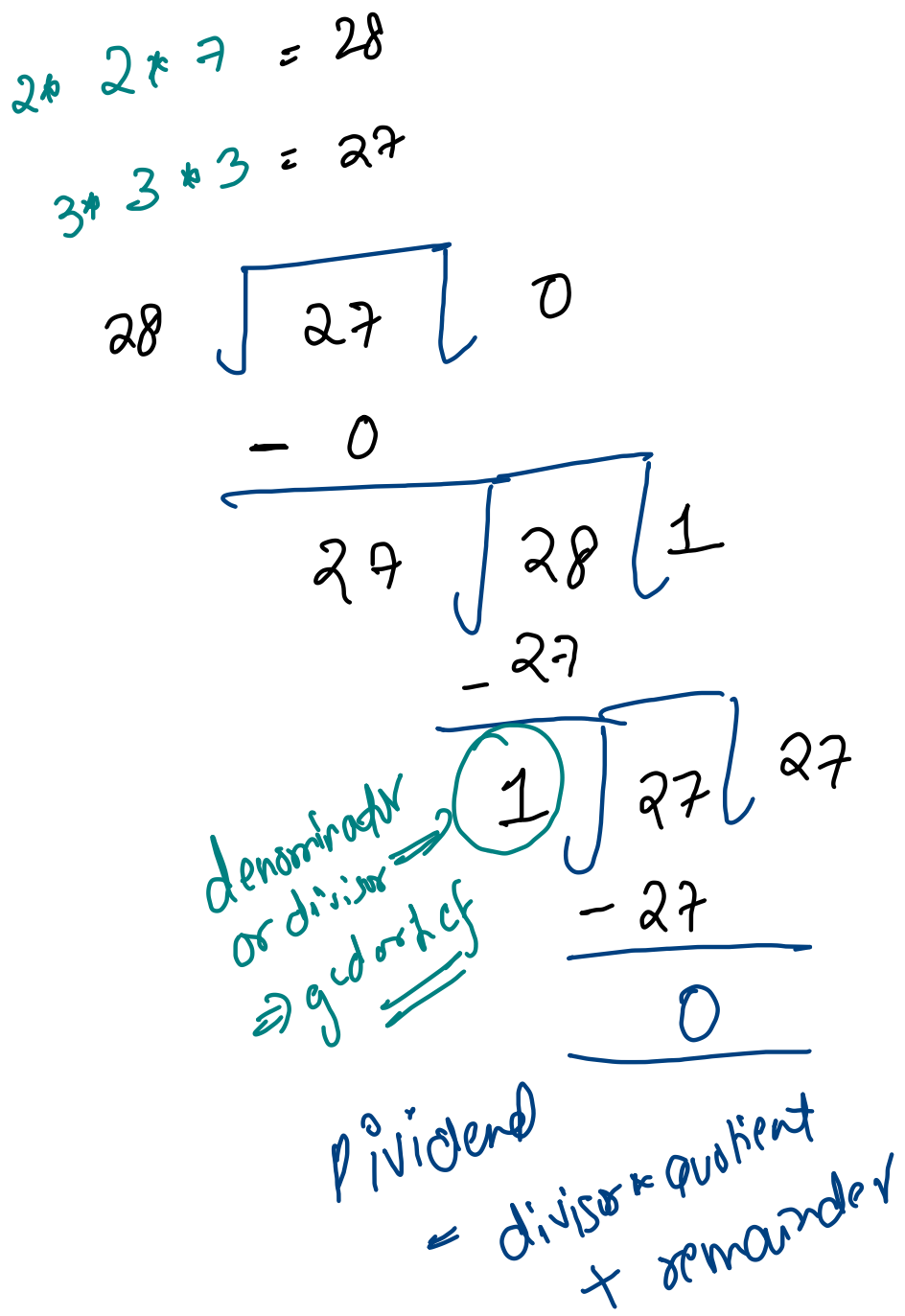
$48 = 2 * 2 * 2 * 2 * 3$

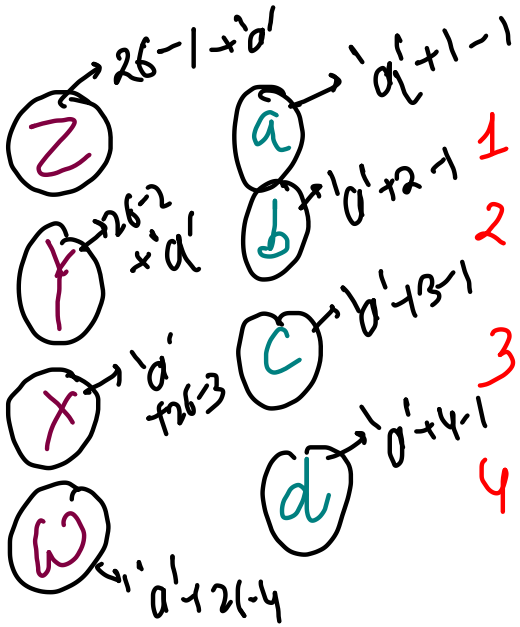
den = 20, num = 48, rem = 8

num = den = 20

den = rem = 8

```java
public class Solution {
    public static int gcd(int a, int b){
        int numerator = a, denominator = b;

        while(numerator % denominator != 0){
            int remainder = numerator % denominator;

            numerator = denominator;
            denominator = remainder;
        }

        return denominator;
    }

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int a = scn.nextInt();
        int b = scn.nextInt();
        System.out.println(gcd(a, b));
    }
}
```





```
char('26-idx+'a')
char('a'+idx-1)
idx
```