Getting Started

Java programming basics

- Boiler plate
- Input (Scanner)
- Printing in Java (print vs println)
- Datatypes & variables of Range & Size
- operators
  - Arithmetic
  - Comparison
  - Logical

- Character datatype (Ascii chart) & Strings
- Comments
- Typecasting & Promotion

Java programming basics

- Conditional Statements
  - if-else
  - nested-if-else
  - if-else-if ladder
- Switch case
- ternary operators ()?:

Java programming basics

- Loops
  - for loop
  - while loop
- break & continue statements
- Functions (Parameter Pass by value & returning values)

Java Programming

Features

→ Object-oriented programming language

→ platform-independent

→ multithreading → concurrency (parallel)

→ Exception handling techniques

→ Collection frameworks

Applications

{ Java → market }

* ① Web Development (webapps)
  - ↳ Springboot (backend)
- ② Graphical User Interface (GUI)
- * ③ Android mobile applications → Java / Koltin
- ④ desktop apps
- ⑤ Data Structures & Algorithms (DSA & OOPS)

Data :→ Raw/Unprocessed Information

College → S_rollno, S_name, S_marks, S_cgpa

Data Structures :→

→ decreasing order of cgpa
→ search all students [⩾8, ⩽9]

efficiently storing data
so that data retrieval is easy!

Algorithms :→ set of instructns/code

Problem-Solving {DSA}

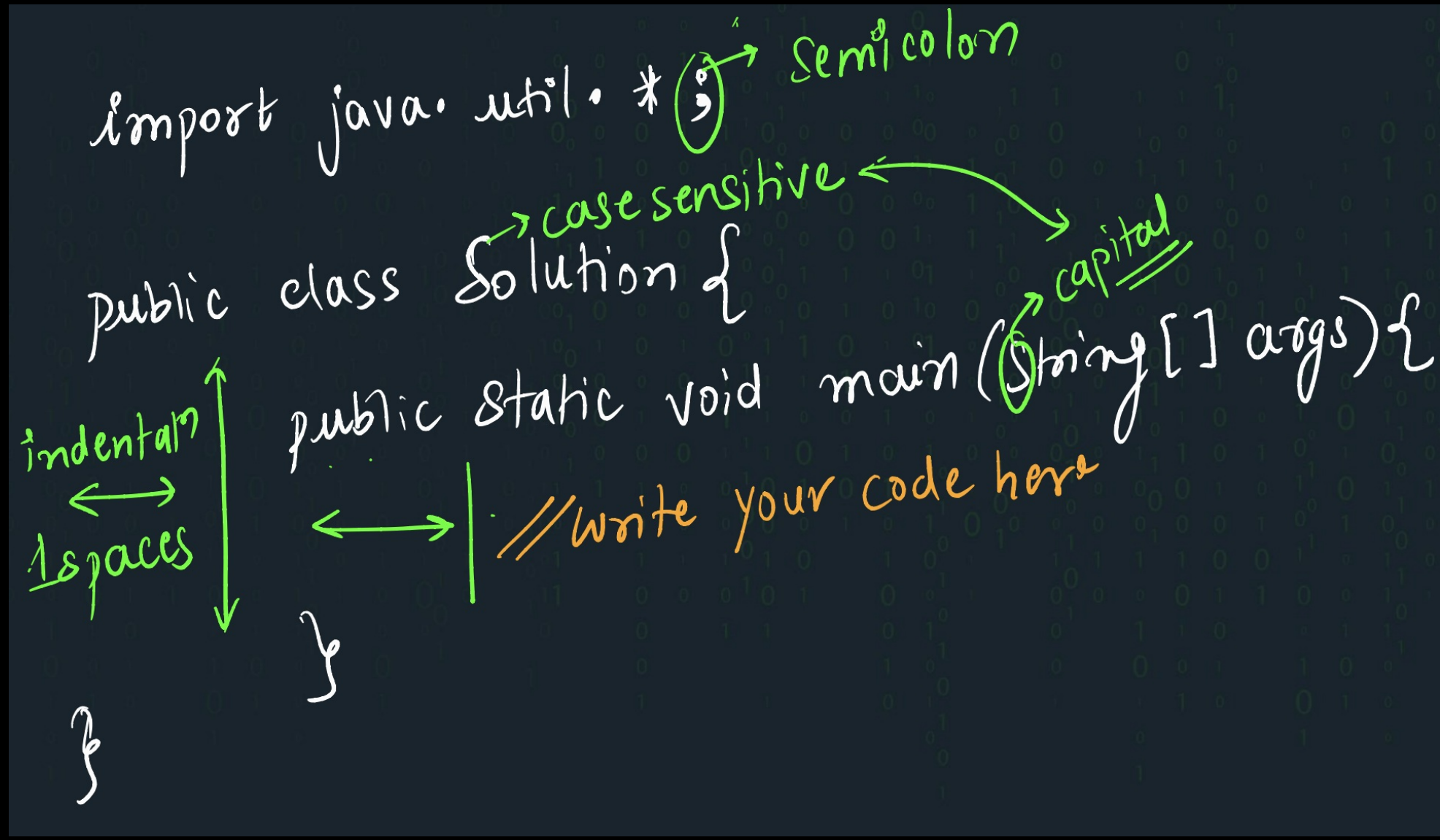
Data Structures

- Arrays & Strings
- Arraylist & StringBuilder
- Linked list
- Stack & queue, Deque
- Hashmap & Heap
- Trees & Graphs

Algorithms

- Two Pointer, Sliding window
- Greedy Algorithms
- Searching & Sorting
- Divide & Conquer
- DP
- Graph Algorithms

Boiler Plate Code



IDE: Integrated Development Environment

→ color scheme or theme

→ auto complete or suggestion

Write your code + debugging the code + running the code + submitting the code

online ide

Web apps :→ hackerrank*, leetcode playground, GFG, etc

local ide

Desktop apps :→ Visual studio code*, intellij, sublime text3, eclispe IDE

Boiler plate code

(case-sensitive) language

Main.java

package

```java
import java.util.*;

public class Main {
    public static void main(String[] args) {

    }
}
```

semicolon

indentation 1tab

indentation

Human Readable Code (Java)

Software

Machine Language

① Printing in Java

System.out.print("helloworld");

```
public static void main(String[] args) {
    System.out.print("Hello World");
    System.out.print("DSA");
    System.out.print("1234@#$%)?");
}
```

```
Finished in 83 ms
Hello WorldDSA1234@#$%)?
```

System.out.println("helloworld");

```
System.out.println("Hello World");
System.out.println("DSA");
```

```
Finished in 82 ms
Hello World
DSA
```

dry run

```java
public static void main(String[] args) {
    System.out.print("Hello World");
    System.out.print("DSA");
    System.out.print("1234@#$%)?");
}
```

Output

Hello WorldDSA1234----

```java
System.out.println("Hello World");
System.out.println("DSA");
```

Output

Hello World

DSA

```java
System.out.println("Hello World");
System.out.print("The Coding Saga");
System.out.println("DSA");
```

Output

Hello world

The Coding SagaDSA

```
System.out.println("10 + 20");
System.out.println(10 + 20);
```

```
Finished in 70 ms
10 + 20
30
```

GFG Hello World (I)

```
class Solution{
    static void printHelloWorld(){
        // code here
        System.out.print("Hello World");
    }
}
```

```
static void printIndividualLine(){
    System.out.println("Geeks");
    System.out.println("for")
    System.out.println("Geeks");
}
```

Compilation error

GFG Hello World (II)

```
static void printIndividualLine(){
    System.out.println("Geeks");
    System.out.println("for");
    System.out.println("Geeks");
}
```

Variables & Datatypes

ram → random access memory

process → run

data → store → ram

primitive → (8)

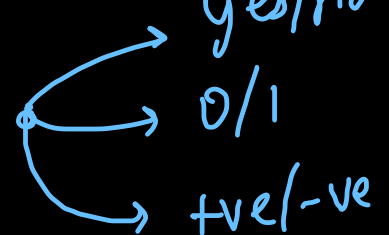
int, char, long, double, float, boolean, short, byte

char → 'a', 'b' ... 'z'
'A' 'B' ... 'Z'
'0' ... '9' '$', '#', '2',

short, byte, int*, long
↳ numerical data
0, 1, 2, 3, -100, 100, ...

float, double*
↳ floating pt data
0.0, 1.5, -2.336, 3.14, ...

boolean*
↳ true/false



true ≠ 1
false ≠ 0

```java
double percentage = 93.4;
System.out.println(percentage);
// System.out.println(marks);

percentage = 95.6; // updation
System.out.println(percentage);
```

```
Finished in 75 ms
93.4
95.6
```

signed

not in Stack

Type	Description	Default	Size	Example Literals	Range of values
boolean	true or false	false	1 bit / 1byte	true, false	true, false
byte	twos-complement integer	0	8 bits 1byte	(none)	-128 to 127
char	Unicode character	\u0000	16 bits (2byte)	'a', '\u0041', '\101', '\\', '\', '\n', 'β'	characters representation of ASCII values 0 to 255
short	twos-complement integer	0	16 bits 2byte	(none)	-32,768 to 32,767
int	twos-complement intger	0	32 bits 4byte	-2,-1,0,1,2	-2,147,483,648 to 2,147,483,647
long	twos-complement integer	0	64 bits 8byte	-2L,-1L,0L,1L,2L	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	IEEE 754 floating point	0.0	32 bits 4byte	1.23e100f , -1.23e-100f , .3f ,3.14F	upto 7 decimal digits
double	IEEE 754 floating point	0.0	64 bits 8byte	1.23456e300d , -123456e-300d , 1e1d	upto 16 decimal digits

① byte
② short
③ int
④ long
① float
② double

Integer.MIN_VALUE ($-\infty$) $\simeq -2*10^9$

Integer.MAX_VALUE ($+\infty$) $\simeq 2*10^9$

byte → 8 bits (1byte) → 2^7 nos -ve, 2^7 nos +ve ⇒ -2^7 to 2^7-1

-128 to 127

short → 16 bits (2byte) → 2^{15} nos -ve, 2^{15} nos +ve ⇒ -2^{15} to $2^{15}-1$

-32768 to 32767

int → 32 bits (4byte) → 2^{31} nos -ve, 2^{31} nos +ve ⇒ -2^{31} to $2^{31}-1$

-2147483648 to 2147483647

long → 64 bits (8byte) → 2^{63} nos -ve, 2^{63} nos +ve ⇒ -2^{63} to $2^{63}-1$

$-9*10^{18}$ to $9*10^{18}$

nibble (4 bits)

0/1 0/1 0/1 0/1 $\Rightarrow 2 \times 2 \times 2 \times 2 = 2^4 = 16$

$\Rightarrow$ 8 nos -ve, 8 nos -ve

$2^4/2 = 2^{4-1} = 2^3$

$2^4/2 = 2^{4-1} = 2^3$

-1, -2, -3, -4, -5, -6, -7, -8

0, 1, 2, 3, 4, 5, 6, 7

```java
float percentage = 93.4f;
System.out.println(percentage);

long inf = 21474836481;
System.out.println(inf);
```

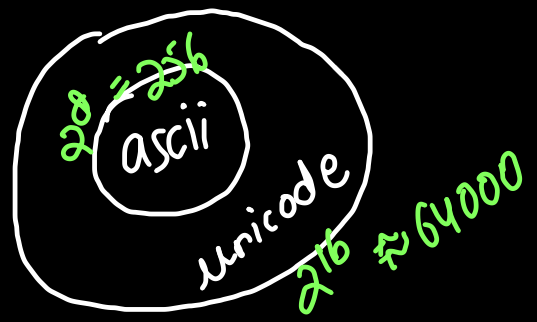
```
Finished in 79 ms
93.4
2147483648
```

Char Datatype

Symbols / letters / digits /

Special escape sequences

unicode chart



ASCII Chart

65 - 90 : 'A' - 'Z'

97 - 122 : 'a' - 'z'

48 - 57 : '0' - '9'

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

```java
char initial = 'a';
System.out.println(initial);

char digit = '0';
System.out.println(digit);

char space = ' ';
System.out.println(space);

char hash = '#';
System.out.println(hash);

char hindi = '\u0917';
System.out.println(hindi);
```

```
Finished in 70 ms
a
0

#
ग
```

Arithmetic operators

binary operators

- a + b ⇒ addition
- a - b ⇒ subtraction
- a / b ⇒ division (quotient)
- a * b ⇒ multiply
- a % b ⇒ modulus (remainder)

unary operator

- a++, ++a ⇒ increment value by 1
- a--, --a ⇒ decrement value by 1

```
Finished in 69 ms
30
10
200
2
```

```java
int a = 20, b = 10;

int c = a + b;
System.out.println(c);

c = a - b;
System.out.println(c);

c = a * b;
System.out.println(c);

c = a / b;
System.out.println(c);
```

```java
System.out.println(3 / 2); // 1.5 -> 1
System.out.println(3 / 4); // 0.75 -> 0
System.out.println(-5 / 2); // -2.5 -> -2

// double division (either num or den or both)
System.out.println(3.0 / 2); // 3.0 / 2 = 1.5
System.out.println(3 / 4.0); // 3 / 4.0 = 0.75
System.out.println(1.0 / 3.0); // 0.3333
```

```java
double d = 3 / 2; // store 1 -> 1.0
System.out.println(d);

double e = 4 / 2; // store 2 -> 2.0
System.out.println(e);
```

```
System.out.println(4 % 2); // 4 = 2 * 2 + 0
System.out.println(3 % 2); // 3 = 2 * 1 + 1

System.out.println(10 % 6); // 10 = 6 * 1 + 4
System.out.println(-10 % 6); // -10 = 6 * -1 + (-4)
```

Property 1

2n+0 even $\%2 = 0$

0, 2, 4, 6, 8

2n+1 odd $\%2 = 1$

1, 3, 5, 7, 9

$[0, 1]$

$a \% b \Rightarrow [0, b-1]$ $a>0, b>0$

$b = 5$

$0\%5 = 0$
$1\%5 = 1$
$2\%5 = 2$
$3\%5 = 3$
$4\%5 = 4$

$5\%5 = 0$
$6\%5 = 1$
$7\%5 = 2$
$8\%5 = 3$
$9\%5 = 4$

$10\%5 = 0$
$11\%5 = 1$
$12\%5 = 2$
$13\%5 = 3$
$14\%5 = 4$

Property 2

extract last digit

12345 % 10 = 5

12345 / 10 = 1234

↳ remove last digit

1234 % 10 = 4

1234 / 10 = 123

123 % 10 = 3

123 / 10 = 12

12 % 10 = 2

12 / 10 = 1

1 % 10 = 1

1 / 10 = 0

```java
int a = 10;
System.out.println(a++); // use then change
System.out.println(a);

System.out.println(++a); // change then use
System.out.println(a);
```

10
11
12
12

a++ : Post increment
use then change

a = 10 → 11

++a : Pre Increment
change then use

a = 11 → 12

int a = 10;

```java
System.out.println(--a); // pre decrement -> change then use
System.out.println(a);

System.out.println(a--); // post decrement -> use then change
System.out.println(a);
```

9
9
9
8

```
System.out.println("10 + 20");
System.out.println(10 + 20);
System.out.println(1.5 + 2.5);
System.out.println("A" + "B");
System.out.println("10" + "20");
```

character → 10 + 20

addition → 30

double addition → 4.0

concatenation → AB

concatenation → 1020

```
System.out.println("10" + "20"); // string + string -> concat
System.out.println(10 + "20"); // int + string -> concat
System.out.println("10" + 20); // string + int -> concat

System.out.println(10 + 20 + "10 + 20" + 10 + 20);
```

⇒ 1020

⇒ 1020

⇒ 1020

int + int → addition → 30

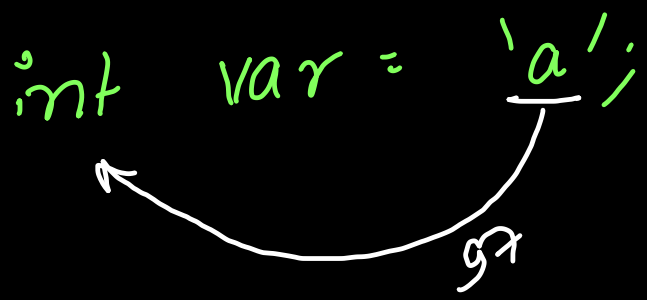
String → concat → int

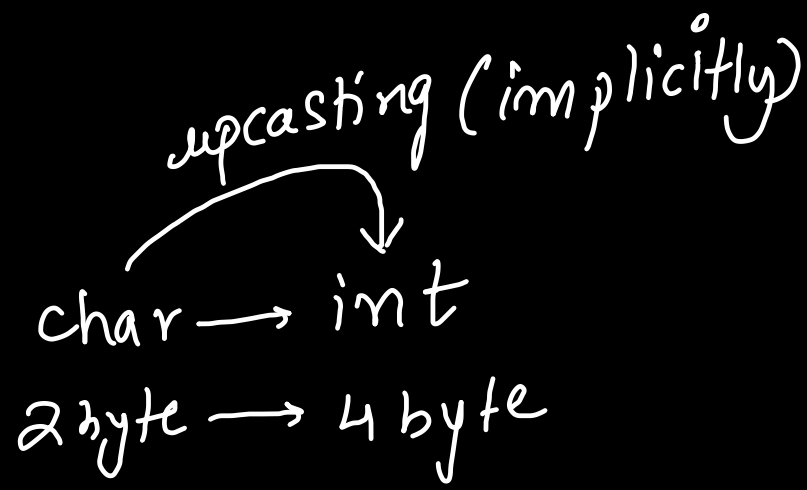
int + String → "3010 + 20"

int → "3010 + 201020"

"3010 + 2010" → String

Type casting






```java
int var1 = 'a'; //97
System.out.println(var1);

int var2 = '0'; // 48
System.out.println(var2);

int var3 = 'Z'; // 90
System.out.println(var3);

int var4 = 'a' + 5; // 97 + 5 = 102
System.out.println(var4);

int var5 = '0' - 5; // 48 - 5 = 43
System.out.println(var5);
```

char + int ⇒ 2byte + 4byte ⇒ int + int
97 + 5 = 102

```java
int var6 = '0' - '0'; // 48 - 48 = 0
System.out.println(var6);

int var7 = 'A' - 'a'; // 65 - 97 = -32
System.out.println(var7);

int var8 = '9' - '0'; // 57 - 48 = 9
System.out.println(var8);

int var9 = '9' - 9; // 57 - 9 = 48
System.out.println(var9);

int var10 = 'd' - 'D'; // 32
System.out.println(var10);
```

```java
int a = 100;
long b = a;
System.out.println(b);

int c = (int)b; // 8 byte -> 4 byte: fit
System.out.println(c);

long d = 99999999999991;
int c = (int)d;
System.out.println(c);
```

4byte 8byte implicit (upcasting or widening)

4byte ← long (downcasting or narrowing explicit)

0 1 1 1 0 1 0 1

overflow ⇓ downcasting

0 1 0 1

```java
int var = 97;
char ch1 = (char)var;
// int (4 byte) -> char (2 byte)
// Implicityly not possible: error
System.out.println(ch1);

int var2 = 48;
char ch2 = (char)var2; // '0'
System.out.println(ch2);

int var3 = 32;
char ch3 = (char)var3; // space
System.out.println(ch3);
```

Inputting Data

Keyboard → (console)

Standard input

```
Scanner scn = new Scanner(System.in);

int marks = scn.nextInt();

double percentage = scn.nextDouble();

long phoneNo = scn.nextLong();
```

Stdout :→ Console / terminal / monitor

```java
System.out.println("Before Exception");
System.out.println(10 / 0);
System.out.println("After Exception");
```

```
Finished in N/A
Before Exception

java.lang.ArithmeticException: / by zero
  at line 200, Main.main
```

runtime error

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

int marks = scn.nextInt();
System.out.println(marks);

double percentage = scn.nextDouble();
System.out.println(percentage);

long phoneNo = scn.nextLong();
System.out.println(phoneNo);

boolean isPass = scn.nextBoolean();
System.out.println(isPass);
```

```
Finished in 109 ms
30
94.5
9319117889
true
```

stdin

```
30
94.5
9319117889
true
```

stdin

```
30
94.5
9319117889
```

```
Finished in N/A
30
94.5
9319117889

java.util.NoSuchElementException
  at line 937, java.base/java.util.Scanner.throwFor
  at line 1594, java.base/java.util.Scanner.next
  at line 1893, java.base/java.util.Scanner.nextBool
  at line 210, Main.main
```

String Data Structure

String → Collection of characters

```
String firstName = "archit";
System.out.println(firstName);
```

```
char letter = scn.next().charAt(0);
System.out.println(letter);
```

→ to take character input

```
Scanner scn = new Scanner(System.in);

String name = scn.next();
System.out.println(name);
```

→ word input

Input
archit aggarwal

Output
archit

```
String name = scn.nextLine();
System.out.println(name);
```

→ Sentence input

Output
archit aggarwal

Relational / Comparison Operators

a & b can be any expression

binary operators
↳ two operands

output : true/false

$a < b$:

$a > b$:

$a <= b$:

$a >= b$:

$a == b$: equality operator

$a != b$

(=) assignment operator
↳ initialize/update

```
System.out.println(5 < 5);
System.out.println(5 > 5);
System.out.println(5 <= 5);
System.out.println(5 >= 5);
System.out.println(5 == 5);
System.out.println(5 != 5);
```

```
Finished in 70 ms
false
false
true
true
true
false
```

T ≠ 1
F ≠ 0

Logical operators

True/false
input → boolean

binary {
- and → a && b → True iff both expressions are true otherwise false
- or → a || b → True if atleast one expression is true otherwise false

unary {
- not → !a → negate answer

!T = F
!F = T

And

a \ b	true	false
true	T && T = true	T && F = false
false	F && T = false	F && F = false

* F && ? = F

or

a \ b	true	false
true	T || T = true	T || F = true
false	F || T = true	F || F = false

* T || ? = T

not

a = true	a = false
!a = !T = F	!a = !F = true

```
System.out.println(10 > 2 || 10 / 0 == 0);
// T || ? = T

System.out.println(10 < 2 && 10 / 0 == 1);
// F && ? = F
```

true

false

(F && F) = F || T = True

(6 == 4 && 24 == 9)

priority / precedence

```
boolean ans1 = (2*3==4 && 6*4==9) || (4>2);
System.out.println(ans1);
```

Arithmetic
1. bracket
2. division/multiplication (L to R)
3. Addition/Subtraction (L to R)
4. Comparison
5. Logical

false && ? = = false

```
boolean ans2 = (4>5) && (3>5 && 80 == 2*40);
System.out.println(ans2);
```

T || ? = True

True && ? = T

```
boolean ans3 = (20*5==100 || 10==10) && (30*2==60 || 40>30);
System.out.println(ans3);
```

= true

```
boolean ans4 = !(30 >= 20 || 40 >= 10);

boolean ans5 = !( (20*4 + 40 >= 100 || 20==10)
                && (3*2<=60 || 4 >= 30));

boolean ans6 = !( 20%3==2 );

boolean ans7 = ( !(40==40) && 80>36);

boolean ans8 = ( !(50>20) || 90>2*45) && ( 30!=2*15);

System.out.println(ans4);
System.out.println(ans5);
System.out.println(ans6);
System.out.println(ans7);
System.out.println(ans8);
```

true || ? = true

!T = false

true

|| ? = true

!(T && T) = !T = F

true || ? = true

!(20%3==2)

2==2

!T = F

!(40==40) && 80>36

!T = F && ? = F

!(50>20) || 90>2*45 && (30!=2*15)

!T = F || F = F && ? = false

false

If-else syntax

control flow statements → conditional statements

```
int marks = scn.nextInt();

if ( marks >= 33) {
        System.out.println( "Pass");
}
```

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

    System.out.println("Before If");
    if(marks >= 33){
        System.out.println("Pass");
    }
    System.out.println("After If");
}
```

Input ⇒ 90

```
Finished in 159 ms
Before If
Pass
After If
```

Input = 20

```
Finished in 138 ms
Before If
After If
```

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

System.out.println("Before If");
if(marks >= 33){
    System.out.println("Pass");
} else {
    System.out.println("Fail");
}
System.out.println("After Else");
```

Input ⇒ 90

```
Finished in 149 ms
Before If
Pass
After If
```

Input ⇒ 20

```
Finished in 110 ms
Before If
Fail
After Else
```

Even & odd ⇒ Geeks for geeks

```java
if(a % 2 == 0){
    // a even, b odd
    System.out.println(a);
    System.out.println(b);
} else {
    // b even, a odd
    System.out.println(b);
    System.out.println(a);
}
```

Example 1:

Input: a = 2, b = 3**Output:** 23

Example 2:

Input: a = 5, b = 2**Output:**25

max^m of two numbers
{ harger number }

int a, b

a = 5 b = 10

output = 10

a = 10, b = 5

output = 10

```
if (a > b) {
    Syso(a);
} else {
    Syso(b);
}
```

check vowel / consonant

input → char → lowercase english alphabet ('a' - 'z')

a → vowel

b → consonant

```
char letter = scn.next().charAt(0);

if(letter == 'a' || letter == 'e' || letter == 'i'
   || letter == 'o' || letter == 'u'){
    System.out.println("Vowel");
} else {
    System.out.println("Consonant");
}
```

If-else-if ladder

```
int marks = scn.nextInt();

if(marks >= 91){
    System.out.println("O");
} else if(marks >= 82){
    System.out.println("A+");
} else if(marks >= 73){
    System.out.println("A");
} else {
    System.out.println("Other");
}
```

Convert marks to grade

91+ → "O"

82+ → "A+"

73+ → "A"

<73 → "other"

```java
int marks = scn.nextInt();

if(marks >= 91){
    System.out.println("O");
}
if(marks >= 82){
    System.out.println("A+");
}
if(marks >= 73){
    System.out.println("A");
}
if(marks > 0) {
    System.out.println("Other");
}
```


```
Finished in 152 ms
O
A+
A
Other
```


```
Finished in 103 ms
A+
A
Other
```


```
Finished in 128 ms
A
Other
```


```
Finished in 120 ms
Other
```

Largest Among Three {a, b, c}

```
int a = scn.nextInt();
int b = scn.nextInt();
int c = scn.nextInt();

1 if(a > b && a > c){
      System.out.println(a);
  } else if(b > c && b > a){
2     System.out.println(b);
3 } else {
      System.out.println(c);
  }
```

a = 15, b = 10, c = 5

1. 15 > 10 && 15 > 5 = (T) ⇒ 15

a = 10, b = 15, c = 5

1. 10 > 15 && ? : F
2. 15 > 5 ⇒ B = 15

a = 10, b = 5, c = 15

1. 10 > 5 && 10 > 15 ⇒ T && F = F
2. 5 > 15 : false
3. (C) ⇒ 15

```java
if(a > b){
    // b cant be maxm
    if(a > c){
        System.out.println(a);
    } else {
        System.out.println(c);
    }
} else {
    // a cant be maxm
    if(b > c){
        System.out.println(b);
    } else {
        System.out.println(c);
    }
}
```

nested if- else

↳ one inside another

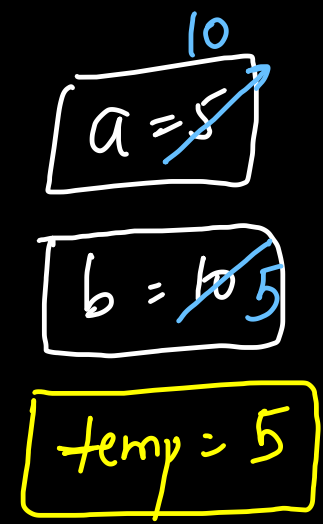
Swap Two Numbers

a = 5, b = 10

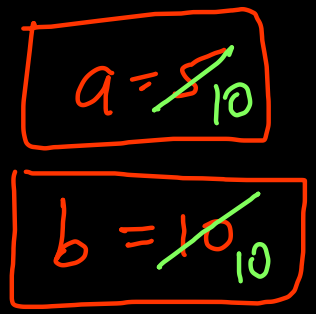
↓ swap

a = 10, b = 5

Approach ① using extra variable




```
int temp = a;
a = b;
b = temp;
```




```
a = b;
b = a;
```

} wrong code

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

    int a = scn.nextInt();
    int b = scn.nextInt();

    // write your code here
    // a = b;
    // b = a;

    // Approach 1
    int temp = a;
    a = b;
    b = temp;

    System.out.println(a);
    System.out.println(b);
}
```

```
Finished in 113 ms
10
5
```

stdin

```
5 10
```

Approach ② using arithmetic operators (without extra variable)

$a = a + b;$

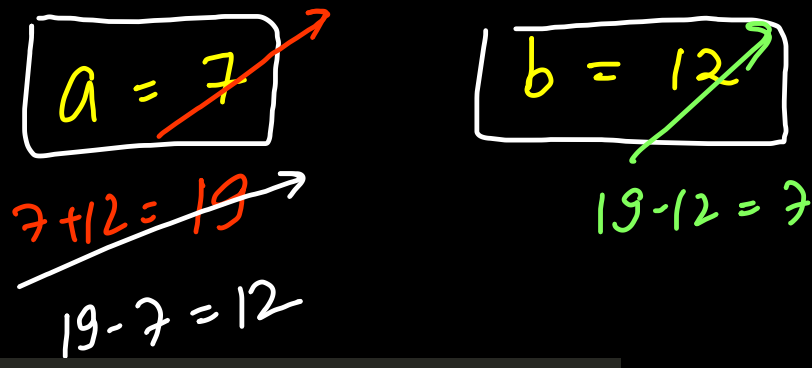
$7 + 12 = 19$

$b = a - b;$

$19 - 12 = 7$

$a = a - b;$

$19 - 7 = 12$




```
// Approach 2
a = a + b;
b = a - b;
a = a - b;
```

Lecture 4

- For loops, while loops, do-while loops
- break, return & continue keywords.
- Switch case
- Ternary operator
- Functions
  - Passing parameters
  - Returning values
  - Pass by value (swap)
  - Function call stack

Loops Problems

1. Print 1 to N, 0 to N-1, N to 1, N-1 to 0
2. Print multiplication table
3. Sum of first N natural nos.
4. Factorial Problem
5. Power Function → Brute force, optimization
6. Nth fibonacci CW, Nth Tribonacci HW
7. Print A to Z
8. Fizz Buzz
9. Running Sum

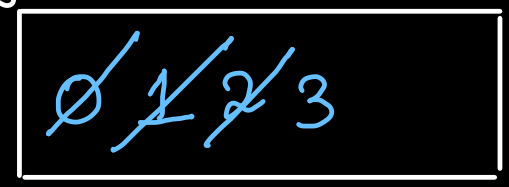
Looping Syntax

- for loop
- while loop
- do-while loop

{for-each loop}

Iteration/Traversal

count




```
for(int count=0 ; count<3 ; count++ ){
    System.out.println(count + " Helloworld");
}
```

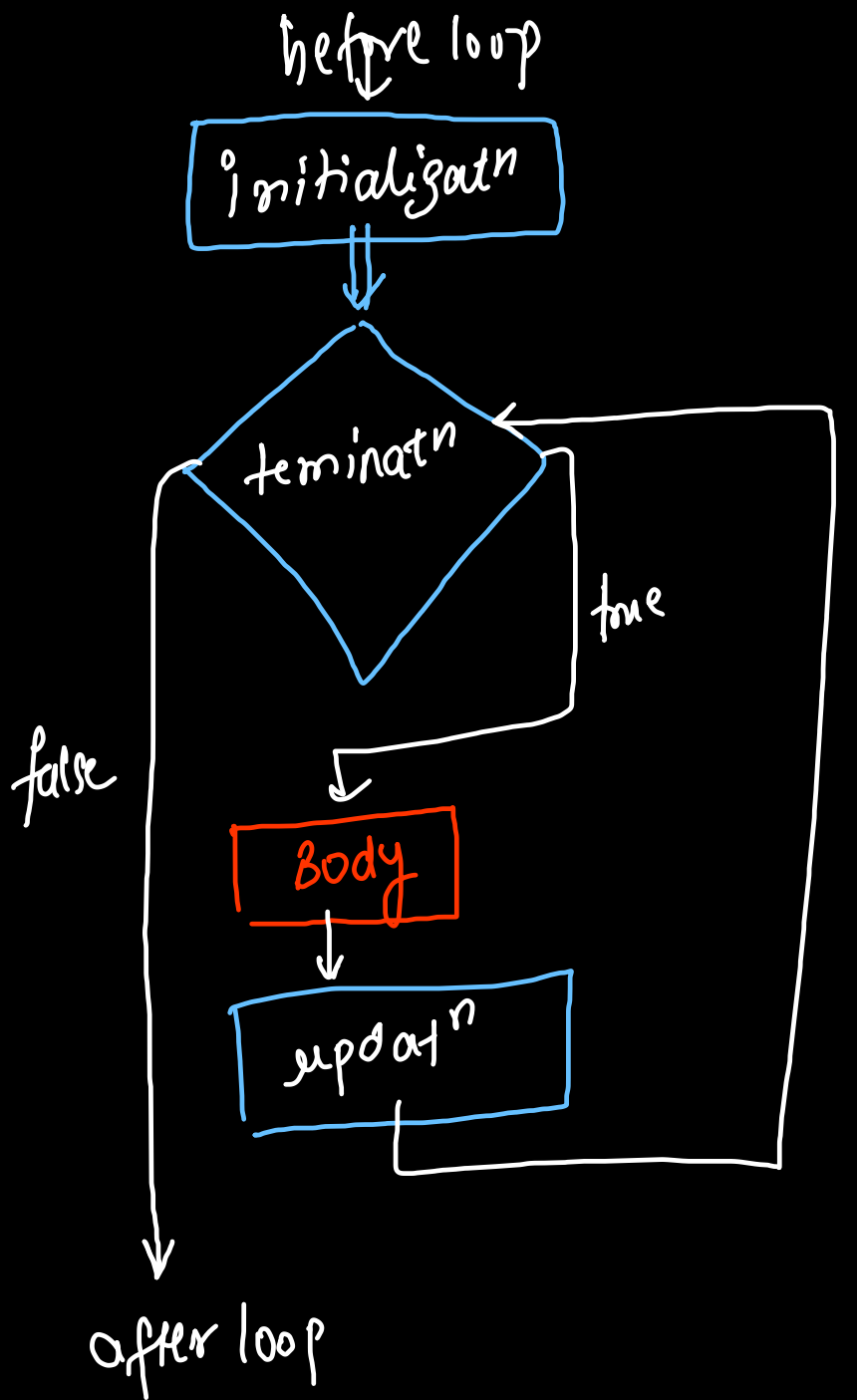
- int count=0 ⇓ initialization
- count<3 ⇓ terminator condition (T&F)
- count++ ⇓ updation

0<3 : 0 Helloworld

1<3 : 1 Helloworld

2<3 : 2 Helloworld

3:3 : false exit the loop




```java
for(int count = 0 ; count < 3 ; count++){
    System.out.println(count + " Hello World");
}
```

```
Finished in 116 ms
0 Hello World
1 Hello World
2 Hello World
```

```java
// 0 Iterations
for(int count = 0; count > 0; count++){
    System.out.println(count + " DSA");
}
```

```
for(int count = 0; count >= 0; count++){
    System.out.println(count + " Coding Saga");
}
```

1* → 2 → 3 → 4

Infinite Loop

runtime error (RE)

or

Time Limit Exceeded (TLE)

count = 0, 1, 2, 3, 4, 5, ...

0 >= 0 : true : 0 CodingSaga

1 >= 0 : true : 1 CodingSaga

2 >= 0 : true : 2 CodingSaga

3 >= 0

CPU crash (abnormal termination)

① Print 1 to N (GFG)

```
for(int idx = 1; idx <= N; idx++){
    System.out.print(idx + " ");
}
```

Output 1 2 3 4

N = 4

idx = 1 2 3 4 5

1 ≤ 4 : true

2 ≤ 4 : true

3 ≤ 4 : true

4 ≤ 4 : true

5 ≤ 4 : false

② Print N to 1 (GFG)

```
for(int idx = N; idx >= 1; idx--){
    System.out.print(idx + " ");
}
```

0-based indexing

(3) Print 0 to N-1

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

or

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

n = 4

{0, 1, 2, 3}

(4) Print N-1 to 0

{3, 2, 1, 0}

```
for(int idx = n-1; idx >= 0; idx--)
```

Print Table of 4

```java
for(int idx = 1; idx <= 10; idx++){
    System.out.println("4x" + idx + "=" + (4 * idx));
}
```

concatenation

idx

4x 1 = 4
4x 2 = 8
4x 3 = 12
4x 4 = 16
4x 5 = 20
4x 6 = 24
4x 7 = 28
4x 8 = 32
4x 9 = 36
4x 10 = 40

Print 'a' to 'z'

```java
public static void main(String[] args) {
    for(char ch = 'a'; ch <= 'z'; ch++){
        System.out.println(ch);
    }
}
```

or

```java
for(int idx = 0; idx < 26; idx++){
    char ch = (char)(idx + 'a');
    System.out.println(ch);
}
```

ch = 'a'

① ch++; (97+1) ch => 'b'

② ch = ch+1; (98+1=99) ch => 'c'

③ ch += 1; ch = 'd'

compound increment

Sum of first N natural nos

① App 1

$$\sum_{i=1}^{N} i = (N*(N+1))/2$$

$n = 5$

sum = 0 + 1 idx = 1 ≤ 5

= 1 + 2 2 ≤ 5

= 3 + 3 3 ≤ 5

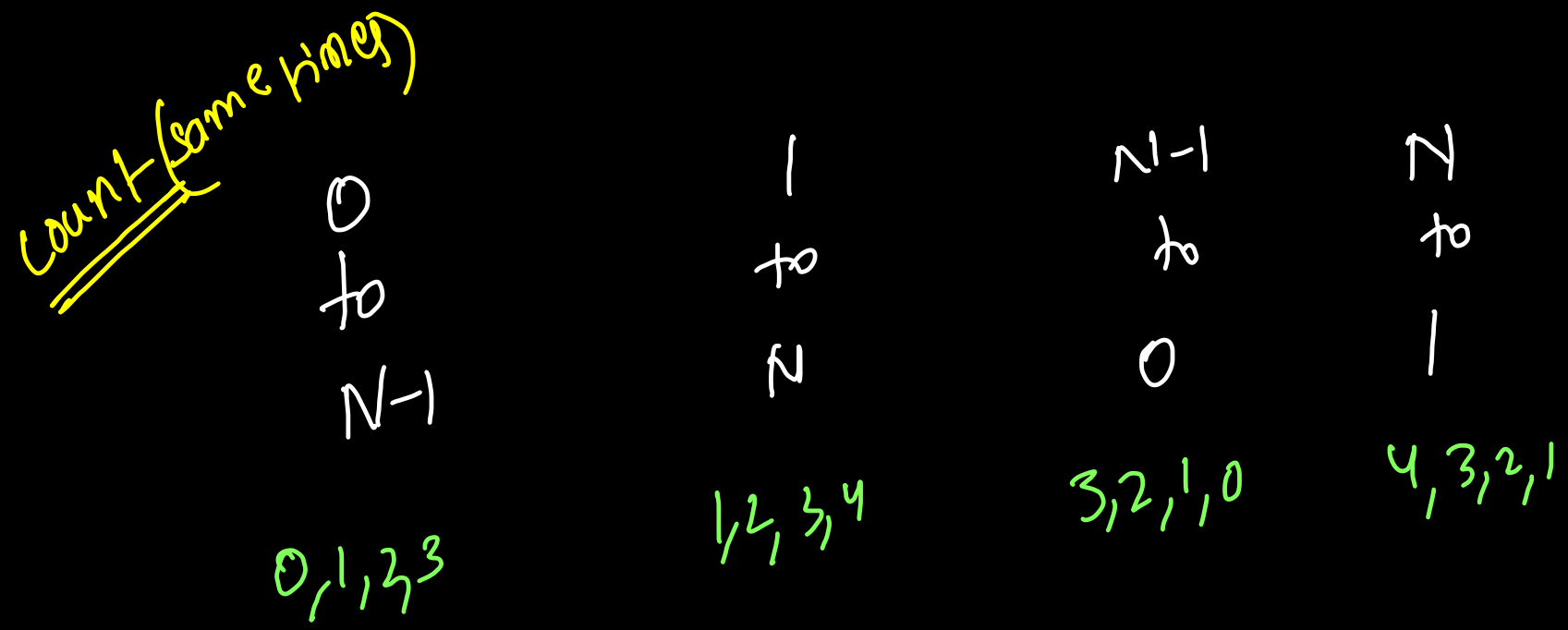
= 6 + 4 4 ≤ 5

= 10 + 5 5 ≤ 5

= 15

② App 2 using loop

```
int sum = 0;
for(int idx = 1; idx ≤ n; idx++){
    sum = sum + idx;
    or
    Sum += idx;
}
System.out.print(sum);
```




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

int sum = 0;
for(int idx = 1; idx <= n; idx++){
    sum += idx; // or sum = sum + idx
}
System.out.println(sum);
```

```java
long sumOfSeries(long N) {
    long sum = 0;

    for(long idx = 1; idx <= N; idx++){
        sum += (idx * idx * idx);
        // or sum = sum + (idx * idx * idx);
    }

    return sum;
}
```

Sum of First N natural no's cube

{GFG}

Factorial problem

$0! = 1$

$1! = 1$

$2! = 1 \times 2 = 2$

$3! = 1 \times 2 \times 3 = 6$

$4! = 1 \times 2 \times 3 \times 4 = 24$

$5! = 1 \times 2 \times 3 \times 4 = 120$

$6! = 1 \times 2 \times 3 \times 4 \times 5 \times 6 = 720$

⋮

```
long prod = 1;
for (int idx = 1; idx <= n; idx++){
    prod *= idx;   (or)   prod = prod * idx;

}
```

```
static long factorial(int N){
    long product = 1;
    for(int idx = 1; idx <= N; idx++){
        product = product * idx;
    }
    return product;
}
```

Power Function or Exponentiation

$a^0 = 1$

int a, b ⇒ $a \wedge b$ ⇒ a * a * ---- a b times

$a^b \leq$ Integer range

$a^b = 2^6 = 2 * 2 * 2 * 2 * 2 * 2 = 64$

$2^1\ 2^2\ 2^3\ 2^4\ 2^5\ 2^6$

(int) Math.pow(a, b);

inbuilt

or

```
int res = 1;
for(int idx = 1; idx <= b; idx++){
    res *= a;
}
return res;
```

$(2.0)^5 = 2.0 \times 2.0 \times 2.0 \times 2.0 \times 2.0 = 32.0$

$(1.2)^4 = 1.2 \times 1.2 \times 1.2 \times 1.2 = 2.0736$

```
double res = 1;
for(int idx = 1; idx <= n; idx++){
    res = res * x;
}
return res;
```

Corner Cases

$(2.0)^{-5} = (1/2.0)^5 = (0.5)^5$

$= 0.5 \times 0.5 \times 0.5 \times 0.5 \times 0.5$

$= 0.03125$

```
public double myPow(double x, int n) {
    if(n < 0){
        x = 1.0 / x;
        n = n * -1;
    }

    double res = 1;
    for(int idx = 1; idx <= n; idx++){
        res = res * x;
    }
    return res;
}
```

Corner case (2) $x = 0.01$, $n = 2147483647 = +\infty$

$0.01 \times 0.01 \times \cdots \infty$ times

Time Limit Exceeded

number will be approaching to 0.0;

Corner case (3)

$x = 1.0$

$n = +\infty$

$(1.0)^{\infty} = 1$

Corner case (4)

$x = -1.0$

$n = +\infty$

$(-1)^{\text{even}} = 1$

$(-1)^{\text{odd}} = -1$

```java
public double myPow(double x, int n) {
    if(x == 1) return 1;
    if(x == -1){
        if(n % 2 == 0) return 1;
        else return -1;
    }

    if(n < 0){
        x = 1.0 / x;
        n = n * -1;
    }

    double res = 1;
    for(int idx = 1; idx <= n && res != 0; idx++){
        res = res * x;
    }
    return res;
}
```

$1^n = 1$

$-1^{even} = +1$

$-1^{odd} = -1$

$x^{-n} = (1/x)^n$

result ≠ 0 ↳ 0 ?

Corner case

$x = -2$

$n = -\infty$

$-(-\infty) = +\infty$

$-(2147483648) = 2147483648$

-2^{31} $\quad$ 2^{31} ✗

$-\infty + 1 = -\infty$

$-\infty - 1 = +\infty$

```java
public double myPow(double x, int b) {
    if(x == 1) return 1;
    if(x == -1){
        if(b % 2 == 0) return 1;
        else return -1;
    }

    long n = b;
    if(n < 0){
        x = 1.0 / x;
        n = n * -1;
    }

    double res = 1;
    for(int idx = 1; idx <= n && res != 0; idx++){
        res = res * x;
    }
    return res;
}
```

$x = -2$, $n = -(2147483648)$

$= -2^{31}$

$-n = +1 \times +2^{31}$

$= 2^{31}$ ↳ integer overflow

long ✗

$= -0$

$(-1/2)^{\infty} = 0$

int: range $\{-2^{31}$ to $2^{31}-1\}$

starting at 10:10 pm

Write a short program that prints each number from 1 to 100 on a new line.

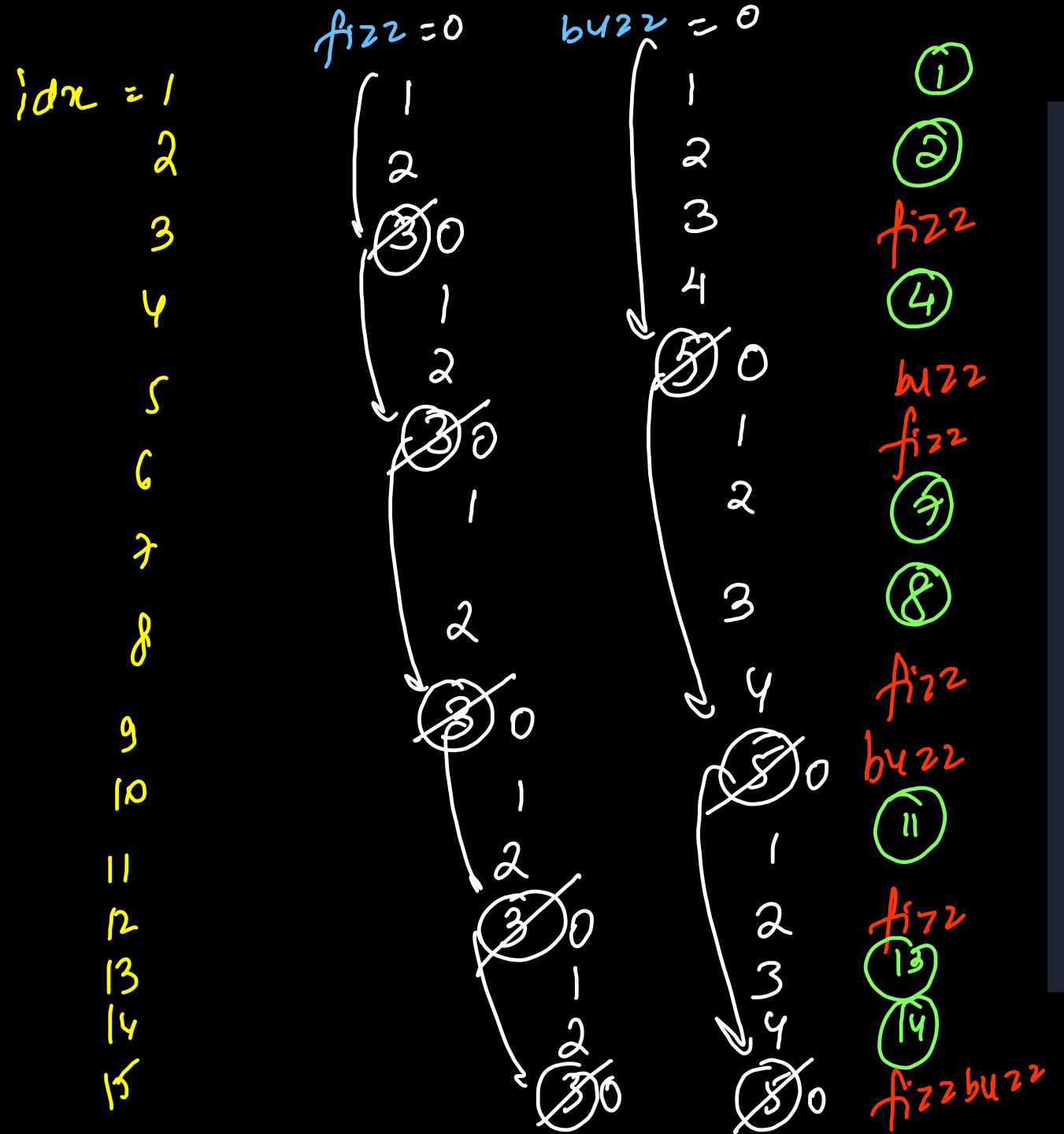
c1 For each multiple of 3, print "Fizz" instead of the number.

c2 For each multiple of 5, print "Buzz" instead of the number.

c3 For numbers which are multiples of both 3 and 5, print "FizzBuzz" instead of the number.

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

11 Fizz (12) 13 14 FizzBuzz (15) ------> 100 no's



w/o /, *, % operator

```java
// Approach 2
int fizz = 0, buzz = 0;

for(int idx = 1; idx <= 100; idx++){
    fizz++; buzz++;

    if(fizz == 3 && buzz == 5){
        System.out.println("FizzBuzz");
        fizz = 0; buzz = 0;
    } else if(fizz == 3){
        System.out.println("Fizz");
        fizz = 0;
    } else if(buzz == 5){
        System.out.println("Buzz");
        buzz = 0;
    } else {
        System.out.println(idx);
    }
}
```

```java
// Approach 1
for(int idx = 1; idx <= 100; idx++){
    if(idx % 3 == 0 && idx % 5 == 0){
        // multiple of 3 & 5
        System.out.println("FizzBuzz");
    } else if(idx % 3 == 0){
        // multiple of 3
        System.out.println("Fizz");
    } else if(idx % 5 == 0){
        // multiple of 5
        System.out.println("Buzz");
    } else {
        // neither 3 nor 5 multiple
        System.out.println(idx);
    }
}
```

optimizam
↳ do not use % , / , *

% → operation
↳ division → remainder

LC 509) Nth Fibonacci no

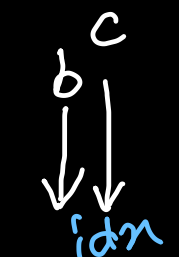
$F(0) = 0$

$F(1) = 1$

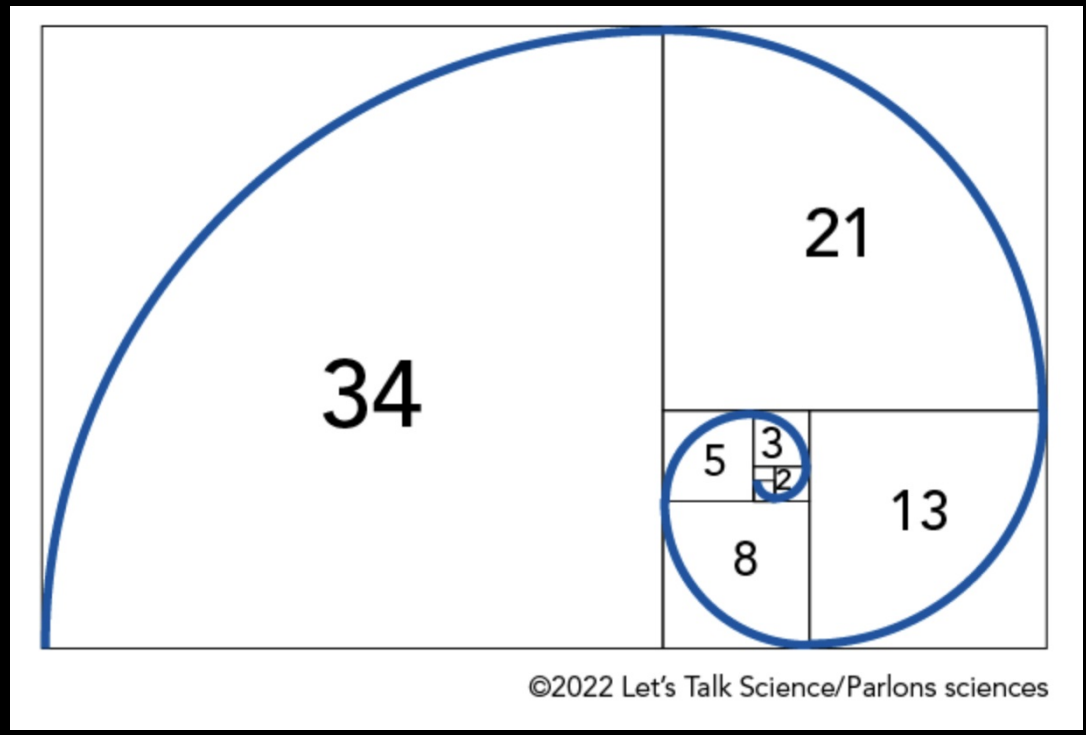
$F(n) = F(n-1) + F(n-2)$ $\quad n \geq 2$

Two Pointer Approach

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



F(0)	F(1)	F(2) idx	F(3) idx	F(4) idx	F(5) idx	F(6) idx	F(7) idx	F(8)
0	1	0+1 =1	1+1 =2	1+2 =3	2+3 =5	3+5 =8	5+8 =13	13+8 =21




```java
public int fib(int n) {
    if(n == 0){return 0;}// f0 = 0
    if(n == 1){return 1;}// f1 = 1

    // fn where n >= 2
    int a = 0, b = 1, c = 1;

    for(int idx = 2; idx <= n; idx++){
        c = a + b; <= fn = fn-1 + fn-2
        a = b;
        b = c; } shifting for new fib calculatn
    }

    return c;
}
```

1137. N-th Tribonacci Number

Easy 3.5K 158

Companies

The Tribonacci sequence T_n is defined as follows:

$T_0 = 0$, $T_1 = 1$, $T_2 = 1$, and $T_{n+3} = T_n + T_{n+1} + T_{n+2}$ for n >= 0.

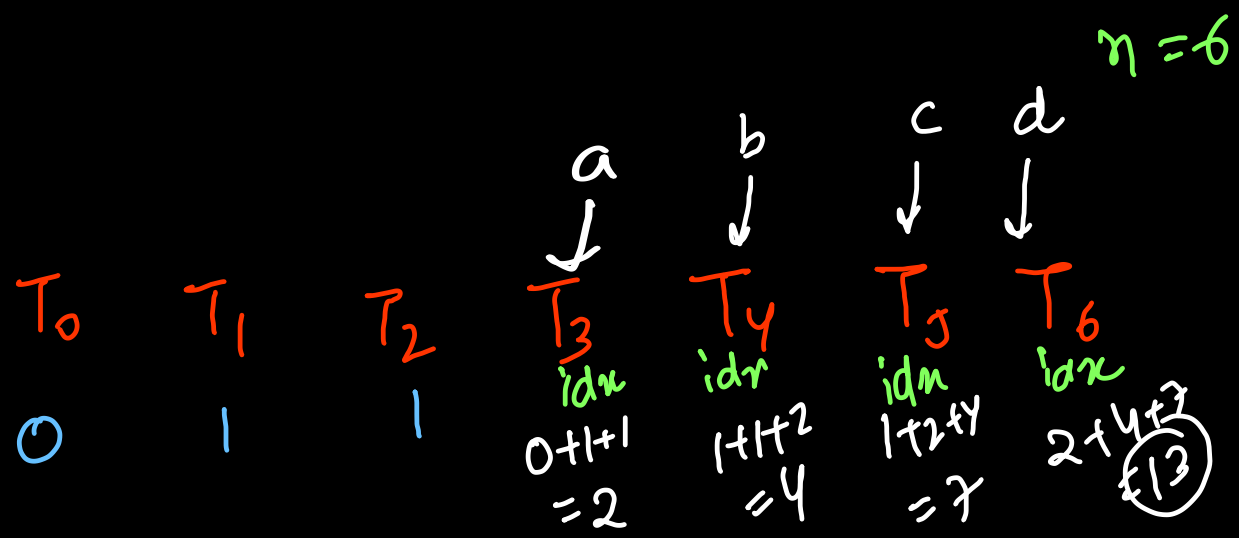
Given `n`, return the value of T_n.

$T_0 = 0$

$T_1 = 1$

$T_2 = 1$

$T_n = T_{n-1} + T_{n-2} + T_{n-3}$ $n \geq 3$




```
d = a+b+c;
a = b;
b = c;
c = d;
```

```java
public int tribonacci(int n) {
    if(n == 0) { return 0; } // T0 = 0
    if(n == 1 || n == 2) { return 1; } // T1 = T2 = 1

    int a = 0, b = 1, c = 1, d = 2;

    for(int idx = 3; idx <= n; idx++){
        d = a + b + c;
        a = b;
        b = c;
        c = d;
    }

    return d;
}
```

Scope → for loop program

```java
public static boolean isPowerofTwo(long n){
    long power = 1; ⇒ implicit gap
    for(; power < n; power = power * 2){
    }

    return (power == n);
}
```

(or)

```java
public static boolean isPowerofTwo(long n){
    long power = 1;
    while(power < n){
        power *= 2;
    }
    return (power == n);
}
```

Check power of 2

64 → true

60 → false

power = 1

1*2

2*2 = 4

4*2 = 8

8*2 = 16

16*2 = 32

32*2 = 64

While loop

for (X ; termination ; X)

X → initialization

X → updation

initialization

while (termination) {

updation

}

Scenario

1. more than 1 variables/ptr
2. conditional based updation
3. variables should be there in memory after loop

Q) Print Pattern

1, 2, 5, 6, 9, 10, 13, 14, 17, 18, 21

odd → +1

even → +3

update omit

```
for(int idx=1; idx<n; ) {
    Syso(idx);
    if(idx%2==0)
        idx+=3;
    else idx++;
}
```

break

```
for (int idx=1; idx<=10; idx++){
    Syso(idx);
    if(idx==5){
        # break;
    }
}
```

⇓

loop terminate

idx = 1, 2, 3, 4, 5 loop terminate

~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~

continue

```
for (int idx=1; idx<=10; idx++){
    if (idx%2==0){ continue; }
    Syso(idx);
}
```

continue ⇒ jump to the next iteration (update)

idx = 1 2 3 4 5

✓ X ✓ X ✓

6 7 8 9 10

X ✓ X ✓ X

```java
for(int idx = 1; idx <= 20; idx++){
    if(idx == 10) break;
    if(idx % 2 == 0) continue;
    System.out.println(idx);
}
```

```
Finished in 77 ms
1
3
5
7
9
```

Digit Traversals

Subtract the Produc | Count Digits | Pract | Reverse Integer - Le | A Number After a D | Armstrong Numbers | PepCoding | Inverse | PepCoding | Rotate

integer $\xrightarrow{n}$ 71542

right to left digits

(ones place to biggest place)

71542 $\xrightarrow{n\%10}$ 2 ✓

$n/10\downarrow$ 7154 $\xrightarrow{n\%10}$ 4 ✓

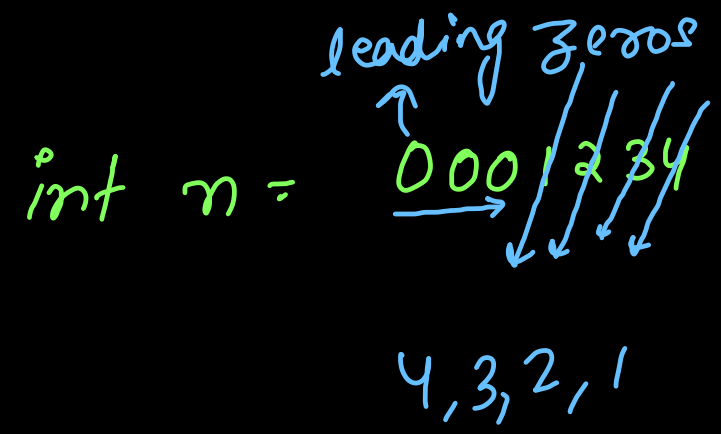
$n/10\downarrow$ 715 $\xrightarrow{n\%10}$ 5 ✓

$n/10\downarrow$ 71 $\xrightarrow{n\%10}$ 1 ✓

$n/10\downarrow$ 7 $\xrightarrow{n\%10}$ 7 ✓

$n/10\downarrow$ 0

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

    while(n != 0){
        int digit = n % 10;
        System.out.println(digit);
        n /= 10; // n = n / 10
    }
}
```



1281. Subtract the Product and Sum of Digits of an Integer

Hint

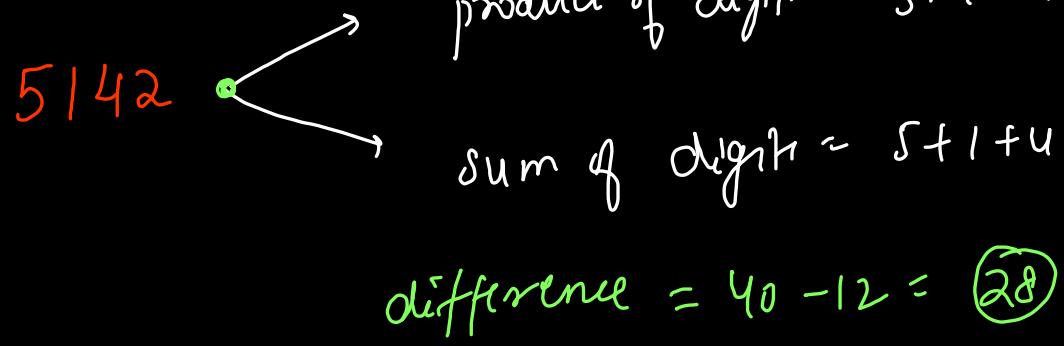
Easy 2.1K 214

Companies

Given an integer number `n`, return the difference between the product of its digits and the sum of its digits.

no of digits ≈ 9

$2^{31} - 1 \approx 2 * 10^9$




```java
public int subtractProductAndSum(int n) {
    int product = 1, sum = 0;
    while(n != 0){
        int digit = n % 10;
        product *= digit; // product = product * digit
        sum += digit; // sum = sum + digit
        n /= 10;
    }

    return (product - sum);
}
```

count total no of digits

71542 count=1

```
int count = 0;

while (n != 0) {
    n = n / 10;
    count++;
}
```

count digits that divides n completely

07154 2

n = 71542 → ✓ (2) → 71542 % 2 == 0 (copy = 71542)

n/10 ↓

7154 → ✗ (4) → 71542 % 4 ≠ 0

n/10 ↓

715 → ✗ (5) → 71542 % 5 ≠ 0

n/10 ↓

71 → ✓ (1) → 71542 % 1 == 0

n/10 ↓

✗ (0) → ignore (division by 0)

7 → ✗ (7) → 71542 % 7 ≠ 0

```java
static int evenlyDivides(int N){
    int copy = N, count = 0;

    while(N != 0){
        int digit = N % 10;
        if(digit > 0 && copy % digit == 0) {
            count++;
        }
        N /= 10;
    }

    return count;
}
```

or

```java
static int evenlyDivides(int N){
    int copy = N, count = 0;

    while(N != 0){
        int digit = N % 10;
        N /= 10;

        if(digit == 0) continue;
        if(copy % digit == 0) {
            count++;
        }
    }

    return count;
}
```

n = 21054 copy = 21054

Count = 0+1+1

21054 —n/10→ 4 21054 % 4 ≠ 0 X

n/10↓

2105 —n%10→ 5 21054 % 5 ≠ 0 X

n/10↓

210 —n%10→ 0 digit ≠ 0 X

n/10↓

21 —n%10→ 1 21054 % 1 == 0 ✓

n/10↓

2 —n%10→ 2 21054 % 2 == 0 ✓

n/10↓

0

HW # Armstrong No.

✓ 371 ⇒ $1^3 + 7^3 + 3^3 = 1 + 343 + 27 = 371$

✓ 153 ⇒ $3^3 + 5^3 + 1^3 = 27 + 125 + 1 = 153$

✗ 372 ⇒ $3^3 + 7^3 + 2^3 = 27 + 343 + 8 \neq 378$

int cube = digit * digit * digit

int cube = (int) Math.pow(digit, 3);

Reverse Integer

Leetcode 7th

eg 1234 → reverse → 4321

Cornercases

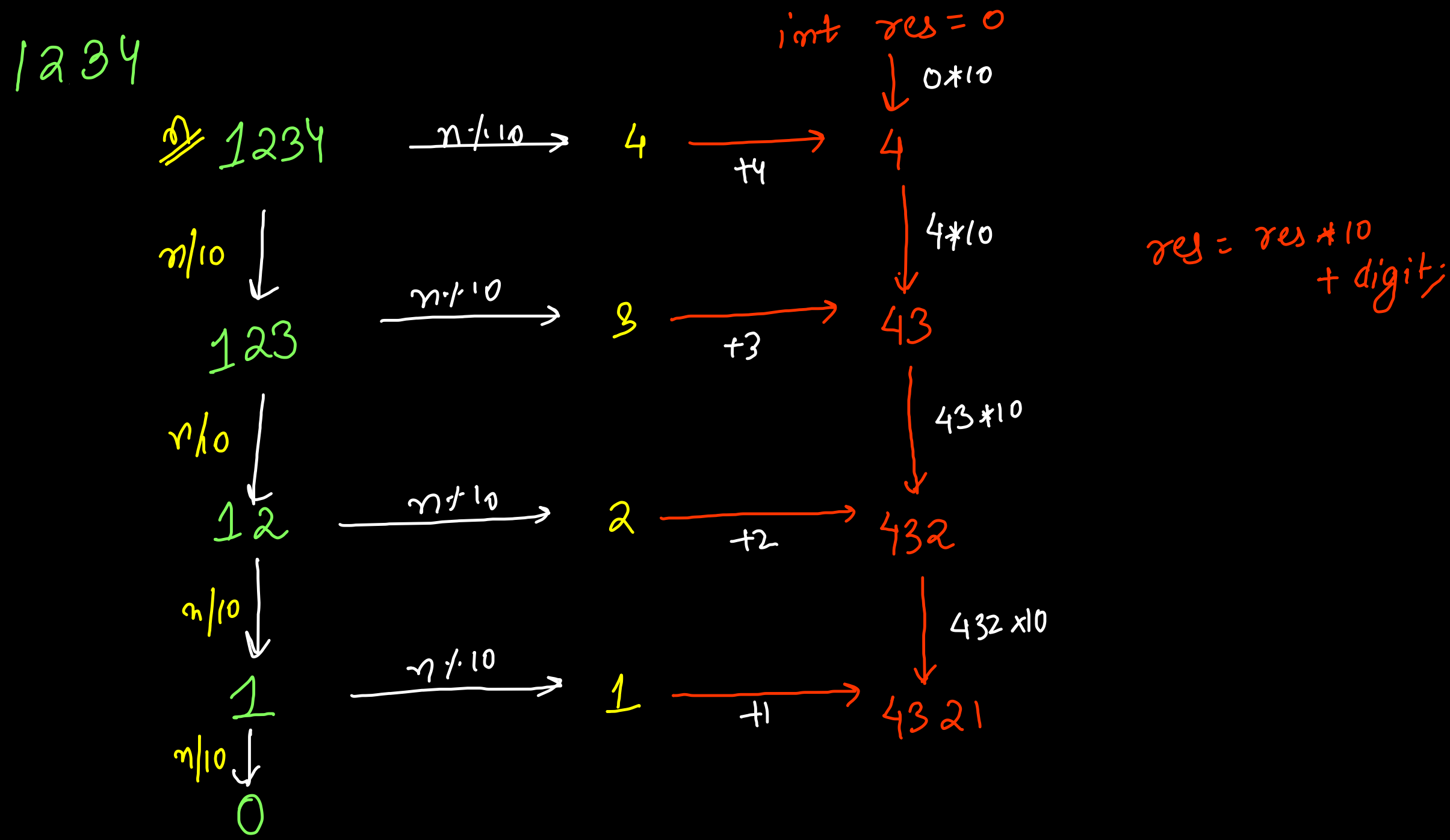
1000 → reverse → 0001 = 1

-123 → reverse → -321

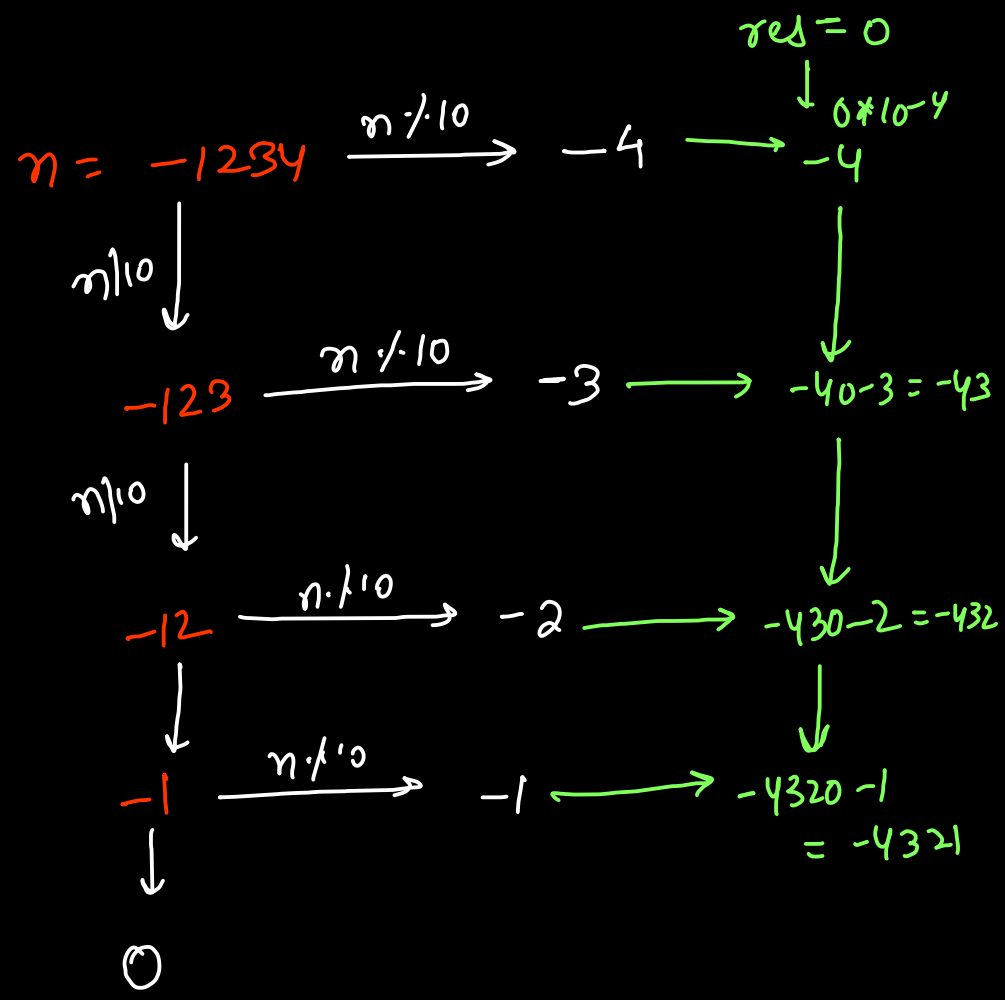
Constraints

Don't

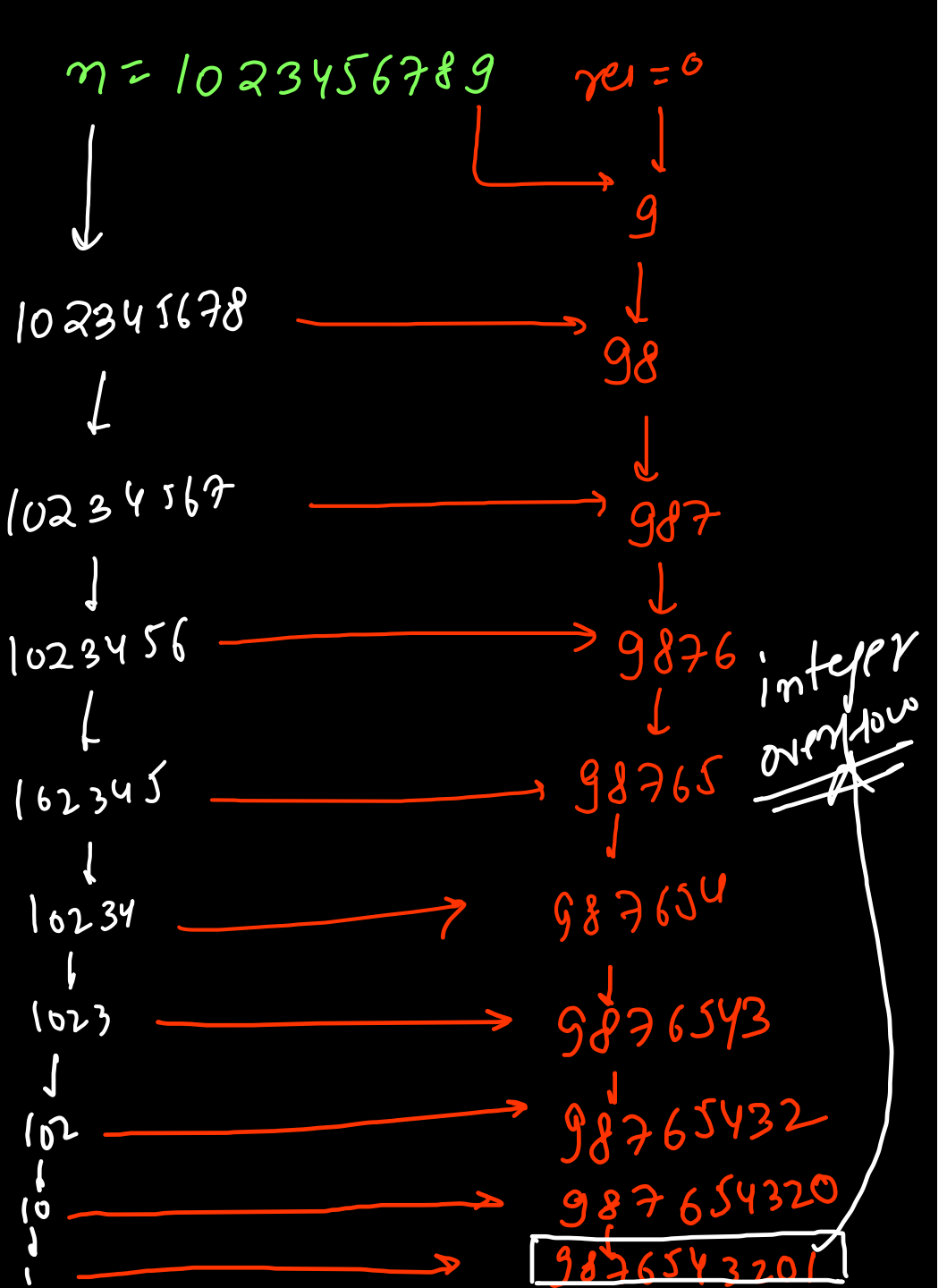
1) Take String
2) Long Datatype
3) Inbuilt Functions


1234
int res = 0
0*10
1234
n%10
4
+4
4
n/10
4*10
123
n%10
3
+3
43
n/10
43*10
12
n%10
2
+2
432
n/10
432 x10
1
n%10
1
+1
4321
n/10
0
res = res *10
+ digit;

```java
public int reverse(int n) {
    int res = 0;

    while(n != 0){
        int digit = n % 10;
        res = res * 10 + digit;
        n /= 10;
    }

    return res;
}
```



```java
public int reverse(int n) {
    int res = 0;

    while(n != 0){
        int digit = n % 10;
        res = res * 10 + digit;
        n /= 10;
    }

    return res;
}
```



```java
public int reverse(int n) {
    int res = 0;

    while(n != 0){
        int digit = n % 10;
        int newres = res * 10 + digit;
        if(newres / 10 != res){
            // integer overflow
            return 0;
        }
        res = newres;
        n /= 10;
    }

    return res;
}
```

to handle negative no's input (eg -1234)

Reverse Integer

Leetcode (7)

to handle integer overflow cases

eg n = 1234567899

2119. A Number After a Double Reversal

Hint

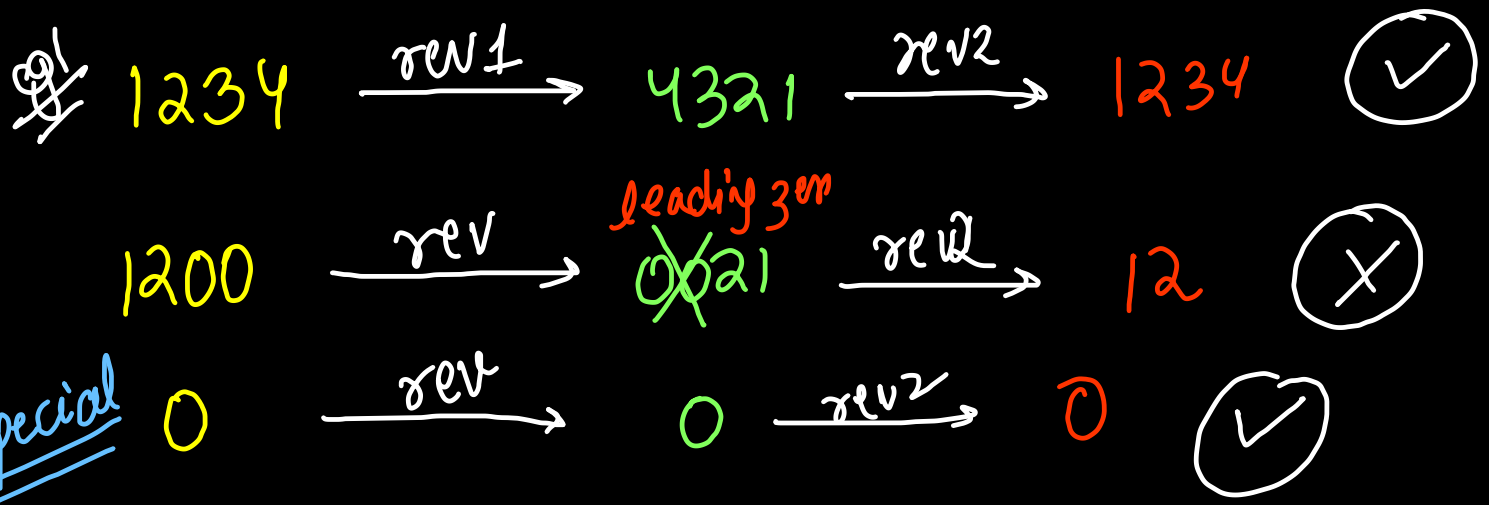
Easy 510 28

Companies

Reversing an integer means to reverse all its digits.

- For example, reversing `2021` gives `1202`. Reversing `12300` gives `321` as the **leading zeros are not retained**.

Given an integer `num`, **reverse** `num` to get `reversed1`, **then reverse** `reversed1` to get `reversed2`. Return `true` *if* `reversed2` *equals* `num`. Otherwise return `false`.



HW Brute force

→ Do 2 reversals

Optimized Approach

→ n == 0 : true

→ trailing zeros (last 0) ⇒ false

→ else true

LC 2119

```java
public boolean isSameAfterReversals(int num) {
    if(num == 0) return true;
    else if(num % 10 == 0) return false; // trailing zeros
    else return true;
}
```

or

```java
public boolean isSameAfterReversals(int num) {
    return (num == 0 || num % 10 > 0);
}
```

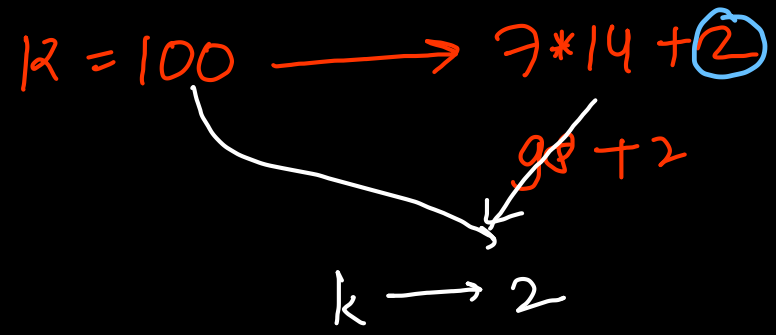
Rotate a No

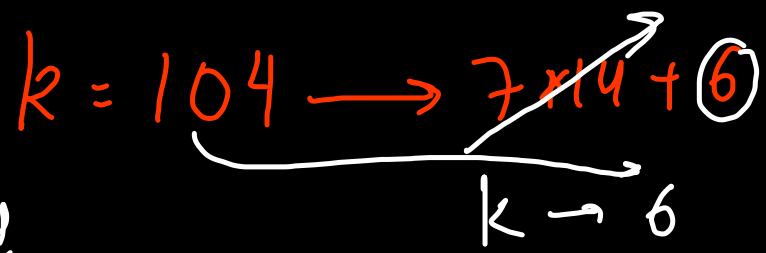
integer

$n = 1234567$

no of digits $d = 7$

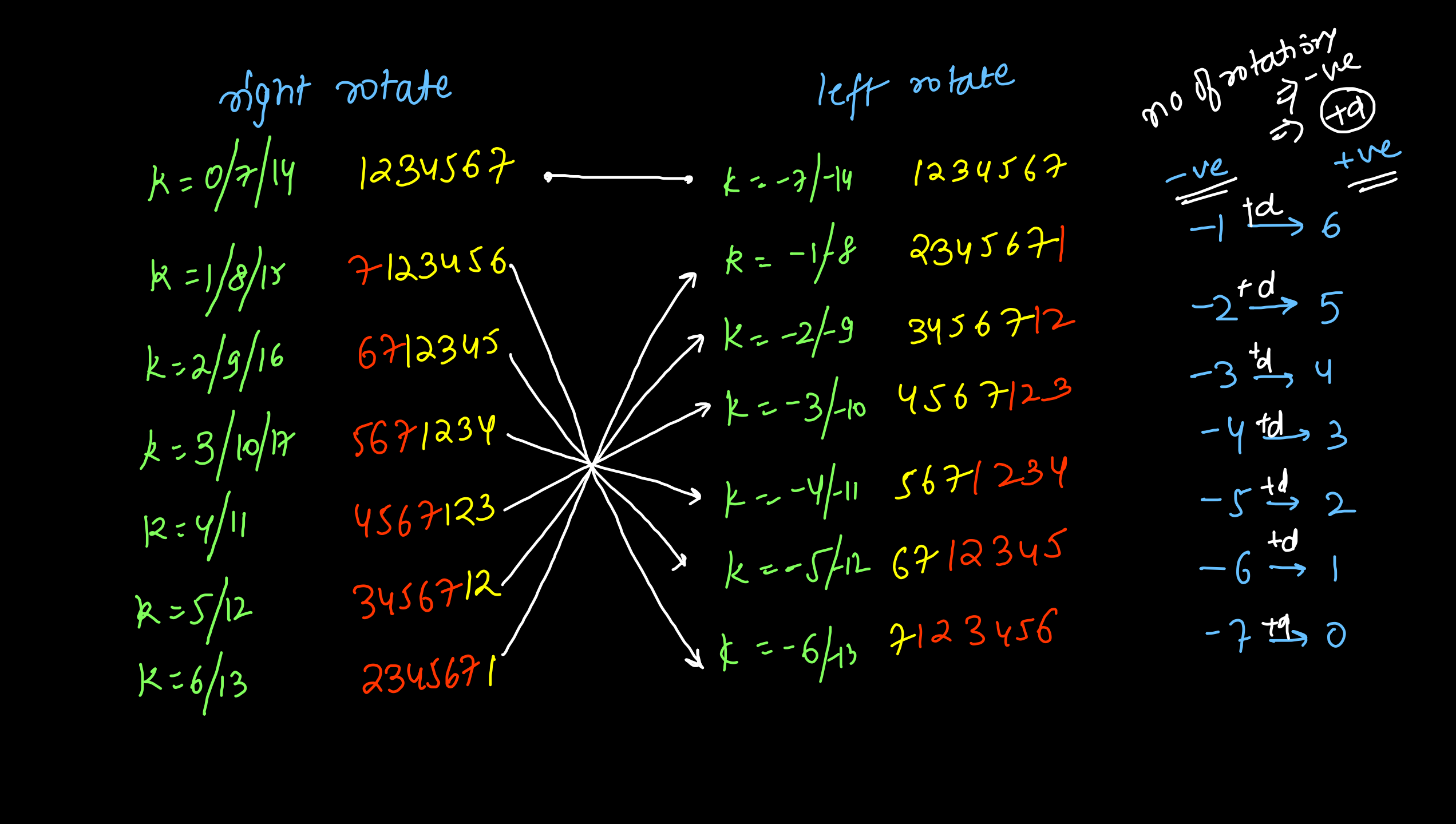
$k = 0/7/14$	1234567
$k = 1/8/15$	7123456
$k = 2/9/16$	6712345
$k = 3/10/17$	5671234
$k = 4/11$	4567123
$k = 5/12$	3456712
$k = 6/13$	2345671





big rotations
Lsmall rotations

```
k = k % d;
```


right rotate
left rotate
k = 0/7/14
1234567
k = 1/8/15
7123456
k = 2/9/16
6712345
k = 3/10/17
5671234
k = 4/11
4567123
k = 5/12
3456712
k = 6/13
2345671
k = -7/-14
1234567
k = -1/-8
2345671
k = -2/-9
3456712
k = -3/-10
4567123
k = -4/-11
5671234
k = -5/-12
6712345
k = -6/-13
7123456
no of rotations
-ve
+ve
-1 +d 6
-2 +d 5
-3 +d 4
-4 +d 3
-5 +d 2
-6 +d 1
-7 +d 0

1234567 $\quad k = [0, 6]$

$k = 3$

red: last k digits = $1234567 \% 10^3 = 567$ $\quad a$

green: first (d-k) digits = $1234567 / 10^3 = 1234$ $\quad b$ (d = 7)

$a = n \% 10^k$

$b = n / 10^k$

5671234

567, 1234 → 5671234

(4) ← 7-3

$567 * 10^4 + 1234 \rightarrow 5671234$

$a * 10^{d-k} + b \Longrightarrow$ resultant

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

int d = 0;
for(int N = n; N != 0; N /= 10){
    d++;
}
// System.out.println(d);

k = k % d; // big k -> small k
// System.out.println(k);

if(k < 0) k += d;
// negative k -> positive k

int a = n % (int)Math.pow(10, k);
System.out.println(a);

int b = n / (int)Math.pow(10, k);
System.out.println(b);

int res = a * (int)Math.pow(10, d - k) + b;
// 567 * 10 ^ 4 + 1234
System.out.println(res);
```

count no of digits

$K = \rightarrow (k \% d + d) \% d$

eg $k = -10, d = 7$

$(-10 \% 7 + 7) \% 7$

$= (-3 + 7) \% 7$

$= 4 \% 7 = 4$

eg $k = 0, d = 7$

$(0 \% 7 + 7) \% 7$

$= (0 + 7) \% 7$

$= 7 \% 7 = 0$

eg $k = 10, d = 7$

$(10 \% 7 + 7) \% 7$

$= (3 + 7) \% 7$

$= 10 \% 7 = 3$

last k digits

rem digits

from the resultant

$|a| = 7$

$\ldots, -21, -14, -7, 0, 7, 14, 21, \ldots \Rightarrow 0$

$\ldots, -20, -13, -6, 1, 8, 15, 22, \ldots \Rightarrow 1 \rightarrow \begin{cases} 7n+1 \\ 7n-6 \end{cases}$

$\ldots, -19, -12, -5, 2, 9, 16, 23, \ldots \Rightarrow 2$

$\ldots, -4, 3, 10, \ldots \Rightarrow 3$

$\ldots, -3, 4, 11, \ldots \Rightarrow 4$

$\ldots, -2, 5, 12, \ldots \Rightarrow 5 \rightarrow \begin{cases} 7n-2 \\ 7n+5 \end{cases}$

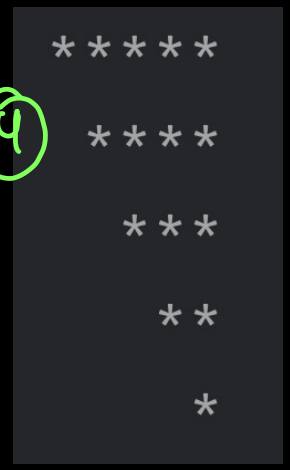
$\ldots, -1, 6, 13, \ldots \Rightarrow 6 \rightarrow \begin{cases} 7n-1 \\ n+6 \end{cases}$

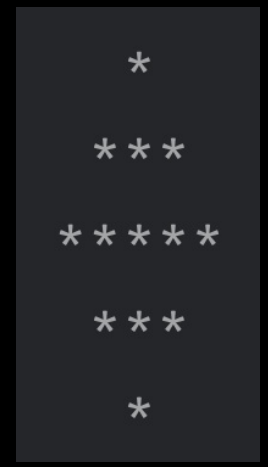
①

②

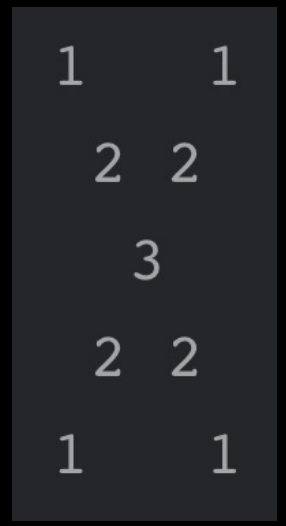
③

④

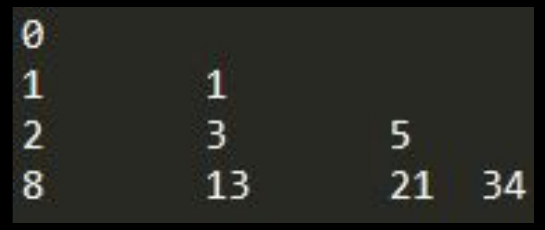


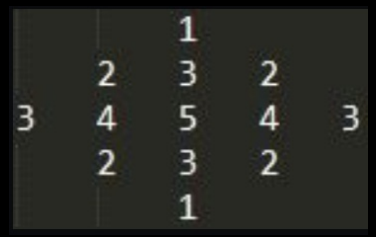


⑤

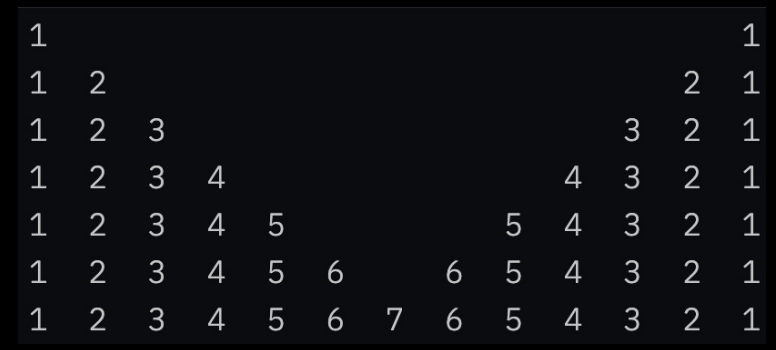


Floyd





⑥



⑦

0

101

21012

Pattern Printing

$n = 5$

nested for loops

```
for(int i=1; i<=n; i++){
    for(int j=1; j<=i; j++){
        Syso.print("*");
    }
    Syso.print(" ");
}
```

```java
static void printPattern(int N){
    for(int i = 1; i <= N; i++){
        for(int j = 1; j <= i; j++){
            System.out.print("*");
        }
        System.out.print(" ");
    }
}
```

n=3

i=1

j=1 1≤1 ✓

j=2 2>1

newspace

i=2

j=1 1≤2 ✓

j=2 2≤2 ✓

j=3 3>2

newspace

i=3

j=1 1≤3 ✓

j=2 2≤3 ✓

j=3 3≤3 ✓

j=4 4>3

new space

i=4>N

Output

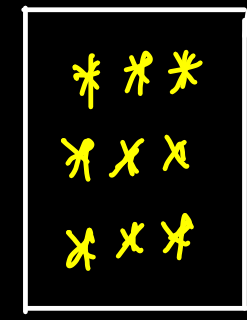
* _ ** _ *** _

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

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

Grid (n * n) square

(hackerrank)

n = 3

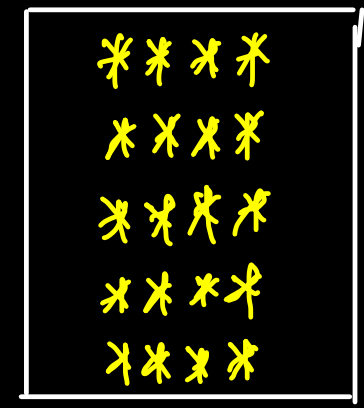


```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int rows = scn.nextInt();
    int cols = scn.nextInt();

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

Grid (rows * cols) rectangular

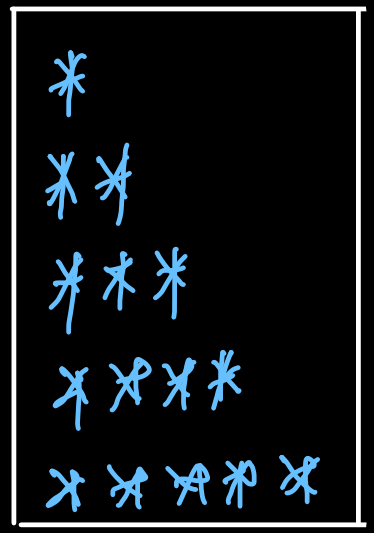
rows = 5

cols = 4



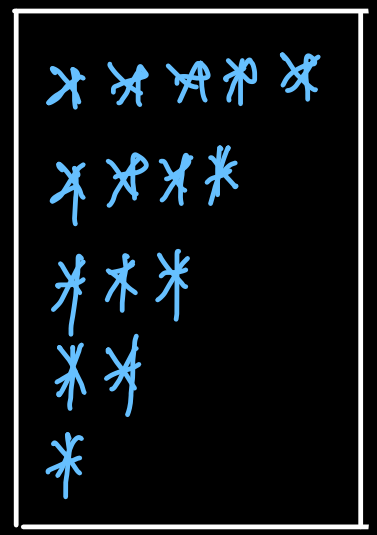
Triangles

n = 5



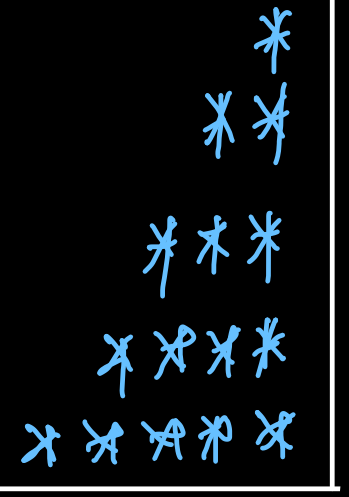
Lower Left triangle

n = 5



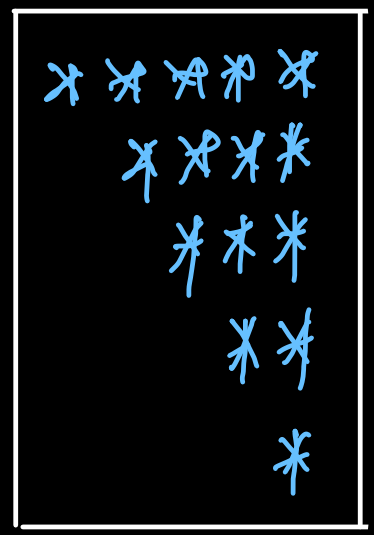
Upper left Triangle

n = 5



Lower Right Triangle

n = 5



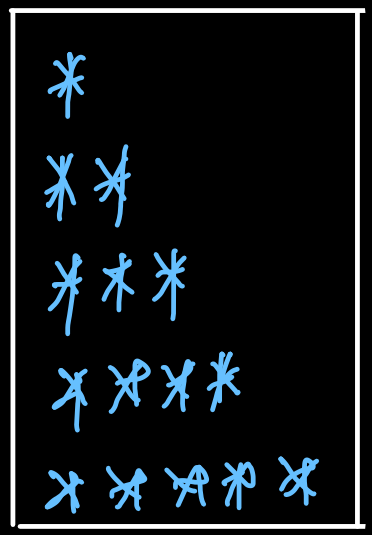
Upper Right Triangle

api1

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

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

Approach ①

n=5



Lower Left triangle

api2

	c1	c2	c3	c4	c5
r1	*	-	-	-	-
r2	*	*	-	-	-
r3	*	*	*	-	-
r4	*	*	*	*	-
r5	*	*	*	*	*

```
for(int r=1; r<=n; r++){
    for(int c=1; c<=n; c++){
        if( c<=r) Syso("* ")
        else Syso(" ");
    }
}
```

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

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

Approach ② Grid-based approach

app3

Stars count = ~~1 2 3 4 5~~

Spaces count = n-1 = ~~4 3 2 1 0~~

	C1	C2	C3	C4	C5
r1	(★	-	-	-	-)
r2	(★	★	-	-	-)
r3	(★	★	★	-	-)
r4	(★	★	★	★	-)
r5	(★	★	★	★	★)

```
int stars = 1, spaces = n-1;
for(int row=1; row<=n; row++)
{
    for(int col=1; col<=stars; col++)
        Syso("* ");
    for(int col=1; col<=spaces; col++)
        Syso(" ");
    Syso.println(); stars++; spaces--;
}
```

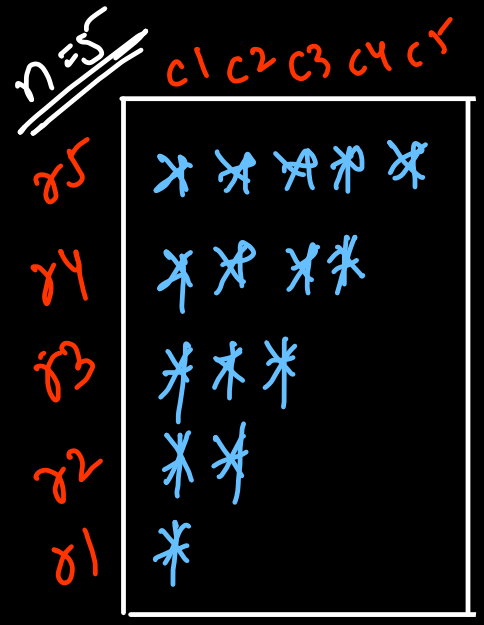
```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();

    int stars = 1, spaces = n - 1;
    for(int row = 1; row <= n; row++){
        for(int count = 1; count <= stars; count++){
            System.out.print("* ");
        }
        for(int count = 1; count <= spaces; count++){
            System.out.print("  ");
        }
        System.out.println();
        stars++; spaces--;
    }
}
```

approach ③

Stars & spaces approach

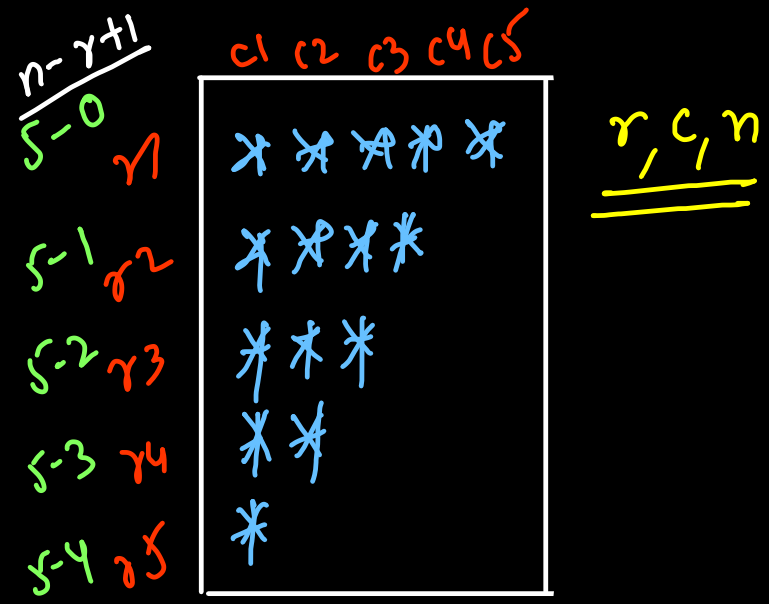
(Decreasing rows)

```java
// Approach 1: Increasing Columns
for (int row = n; row >= 1; row--) {
    for (int col = 1; col <= row; col++) {
        System.out.print("*");
    }
    System.out.println();
}
```

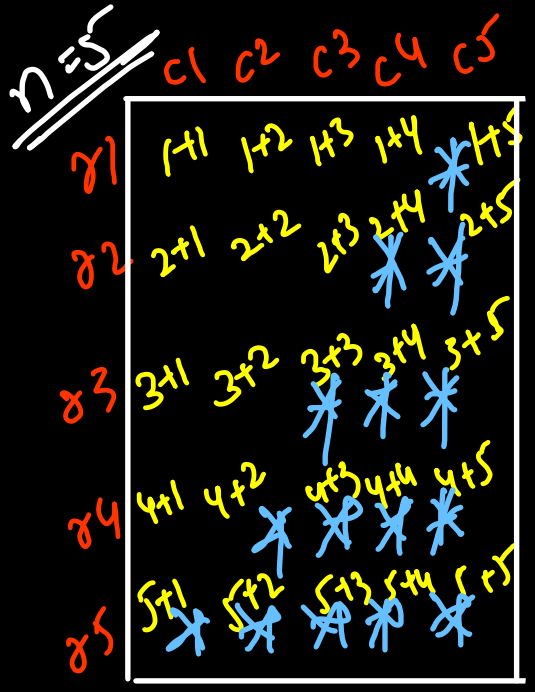


Upper left Triangle

```java
// Approach 2: Grid Based Approach
for (int row = 1; row <= n; row++) {
    for (int col = 1; col <= n; col++) {
        if (n - row + 1 >= col)
            System.out.print("*");
        else
            System.out.print(" ");
    }
    System.out.println();
}
```




```java
// Approach 3: Stars & Spaces Approach
int stars = n, spaces = 0;
for (int row = 1; row <= n; row++) {
    for (int count = 1; count <= stars; count++) {
        System.out.print("*");
    }
    for (int count = 1; count <= spaces; count++) {
        System.out.print(" ");
    }
    System.out.println();
    stars--; spaces++;
}
```



Lower Right Triangle

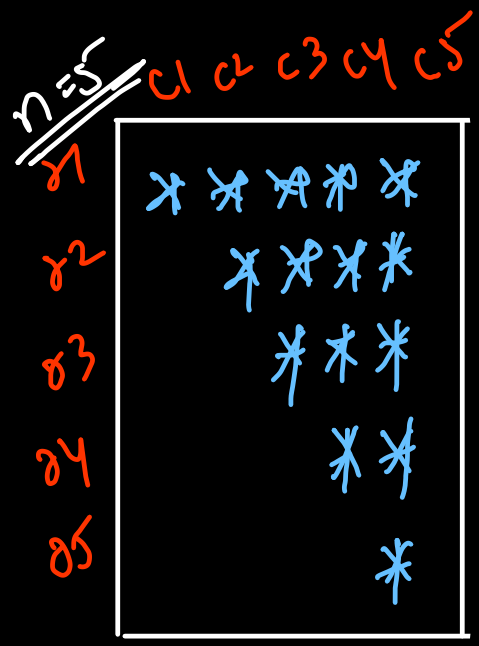
Since spaces are before stars, approach 1 cannot be applied.

```java
// Approach 2: Grid Based Approach
for (int row = 1; row <= n; row++) {
    for (int col = 1; col <= n; col++) {
        if (row + col > n)
            System.out.print("*");
        else
            System.out.print(" ");
    }
    System.out.println();
}
```

```java
// Approach 3: Stars & Spaces Approach
int stars = 1, spaces = n - 1;
for (int row = 1; row <= n; row++) {
    for (int count = 1; count <= spaces; count++) {
        System.out.print(" ");
    }
    for (int count = 1; count <= stars; count++) {
        System.out.print("*");
    }
    System.out.println();
    stars++; spaces--;
}
```

Codestudio

Homework Problem



Upper Right Triangle

app1 :→

```java
// Approach 2: Grid Based Approach
for (int row = 1; row <= n; row++) {
    for (int col = 1; col <= n; col++) {
        if (n - row + 1 >= col)
            System.out.print("*");
        else
            System.out.print(" ");
    }
    System.out.println();
}
```

col >= row

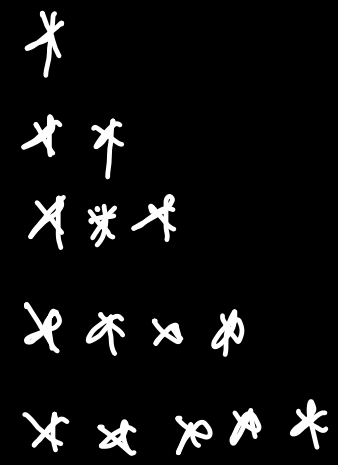
app2 :→

interchange

```java
// Approach 3: Stars & Spaces Approach
int stars = n, spaces = 0;
for (int row = 1; row <= n; row++) {
    for (int count = 1; count <= stars; count++) {
        System.out.print("*");
    }
    for (int count = 1; count <= spaces; count++) {
        System.out.print(" ");
    }
    System.out.println();
    stars--; spaces++;
}
```

Floyd's Triangle

① Structure → Lower Left



n: 5

```
1
23
345
4567
56789
```

```
for(int row = 1; row <= n; row++){
    int val = row;
    for(int col = 1; col <= row; col++){
        System.out.print(val);
        val++;
    }
    System.out.println();
}
```

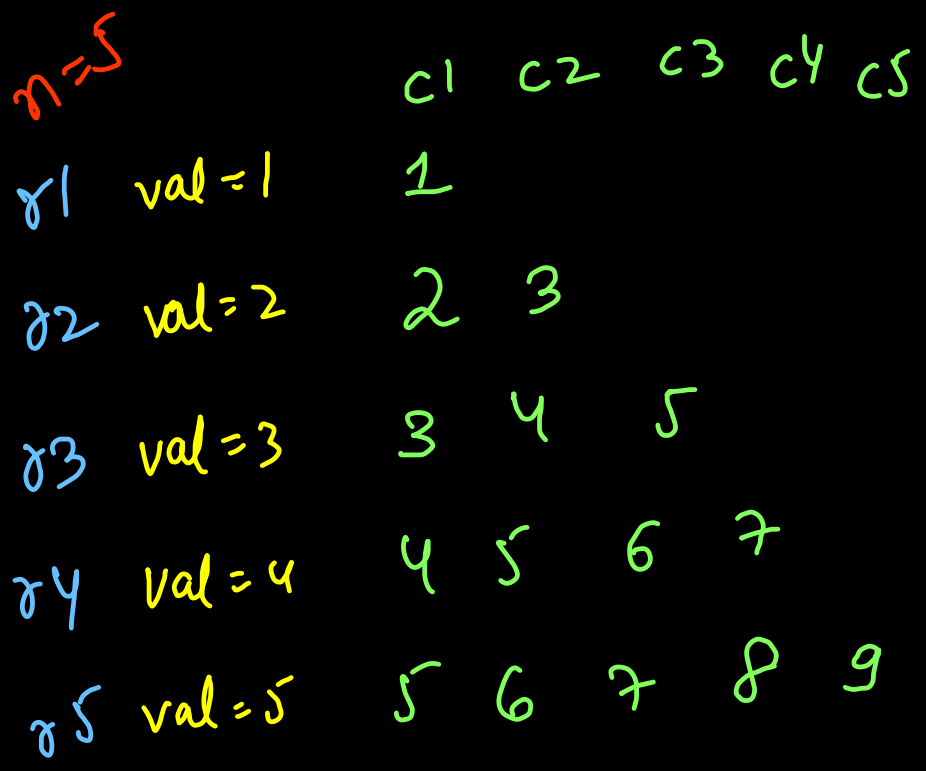
② Convert * to numbers

```
    c1 c2 c3 c4 c5
r1  1
r2  1 2
r3  1 2 3
r4  1 2 3 4
r5  1 2 3 4 5
```

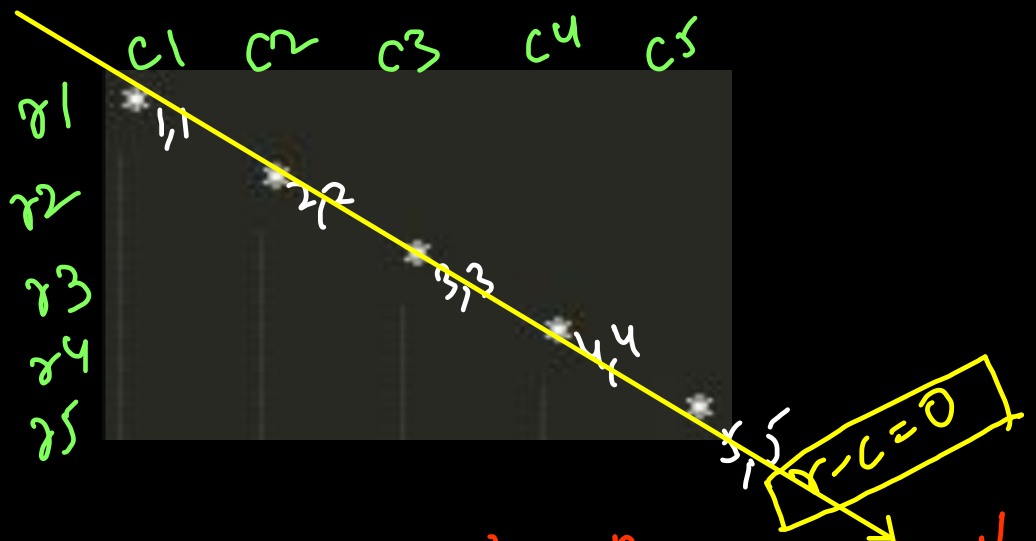

```
    c1 c2 c3 c4 c5
r1  1
r2  23
r3  345
r4  4567
r5  56789
```

```java
for(int row = 1; row <= n; row++){
    int val = 1; row
    for(int col = 1; col <= row; col++){
        System.out.print(val);
        val++;
    }
    System.out.println();
}
```

```java
for(int row = 1; row <= n; row++){
    int val = row;
    for(int col = 1; col <= row; col++){
        System.out.print(val);
        val++;
    }
    System.out.println();
}
```

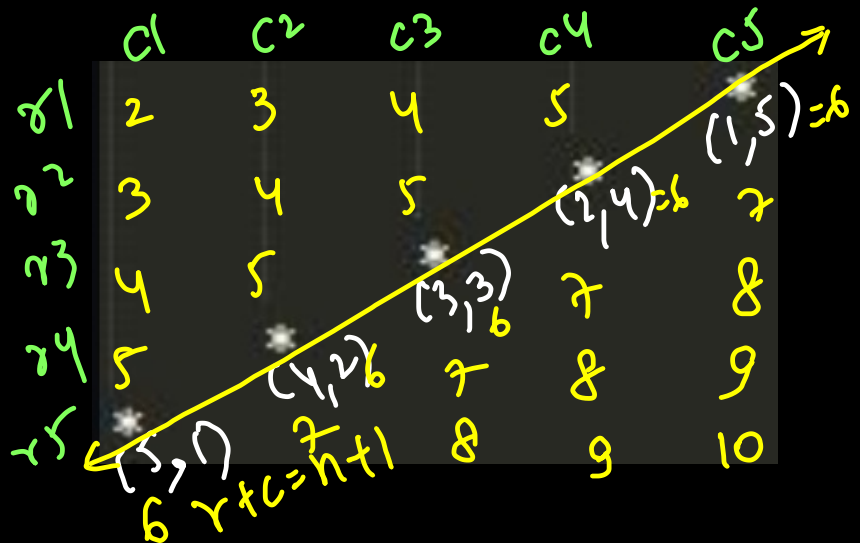


Top Left to Bottom Right Diagonal



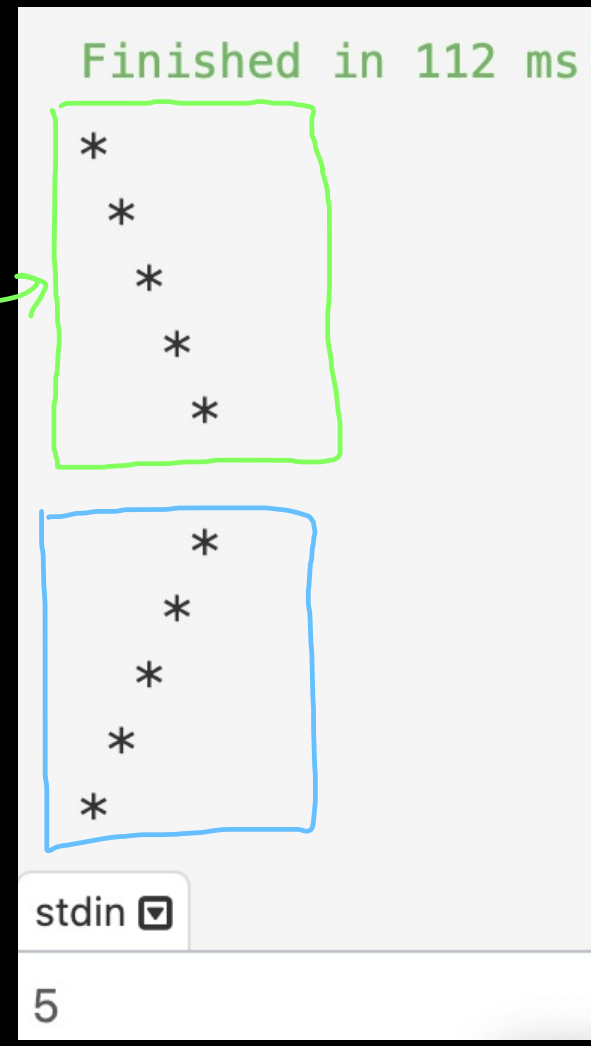

```
for (int r=1; r<=n; r++){
  for (int c=1; c<=n; c++){
    if (r==c) Syso("*");
    else Syso(" ");
  }
  Syso.println();
}
```

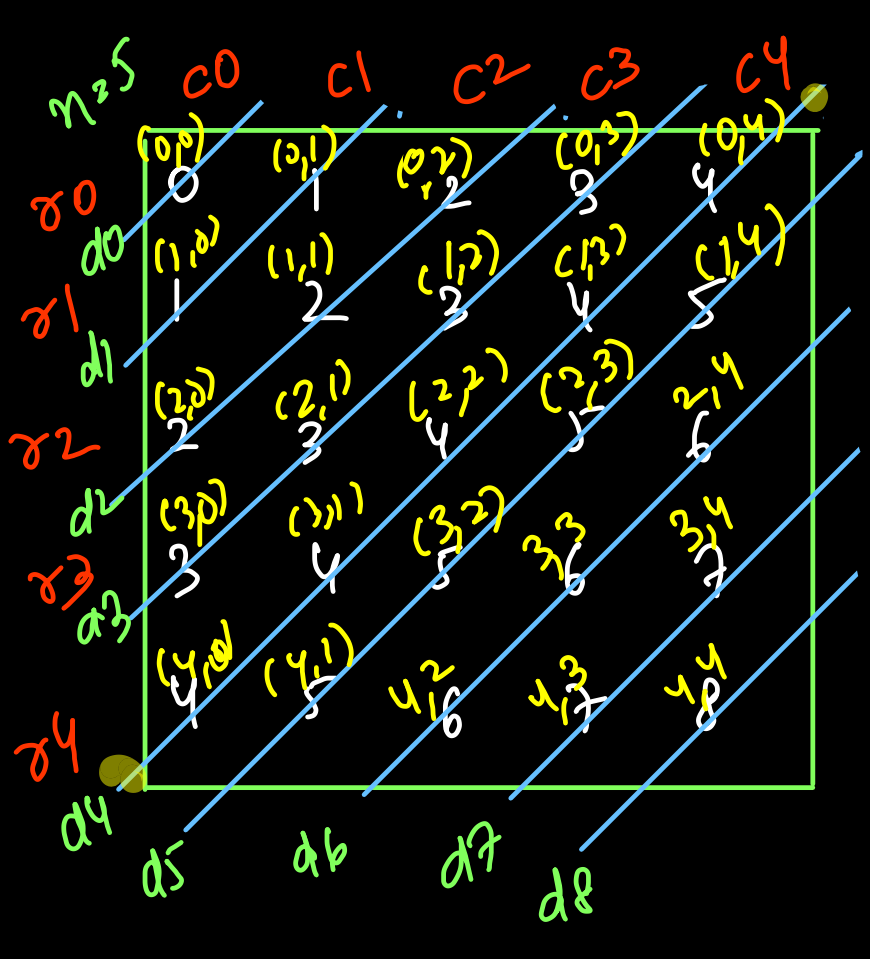
Top Right to Bottom Left Diagonal



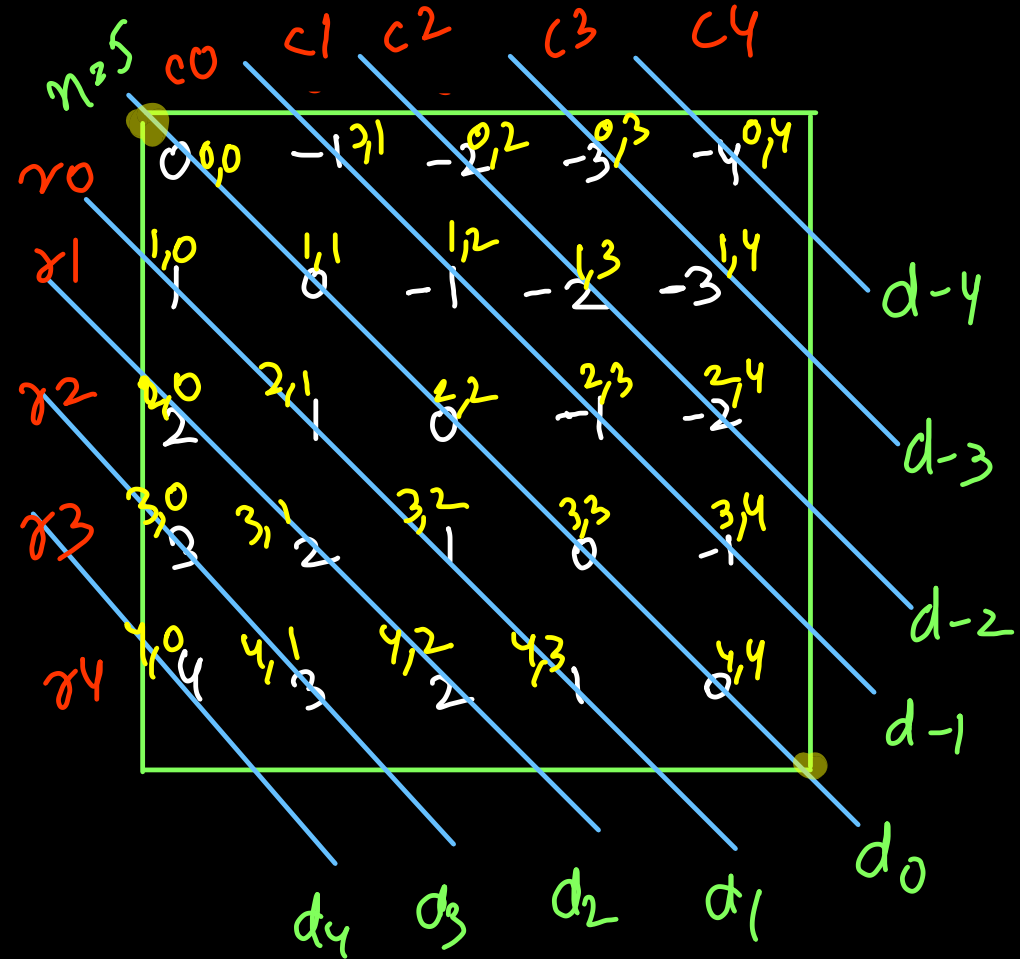

```
for (int r=1; r<=n; r++){
  for (int c=1; c<=n; c++){
    if (r+c==n+1) Syso("*");
    else Syso(" ");
  }
  Syso.println();
}
```

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

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

    System.out.println();

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





row + col



row - col

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

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

System.out.println();

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

Finished in 144 ms

```
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
```

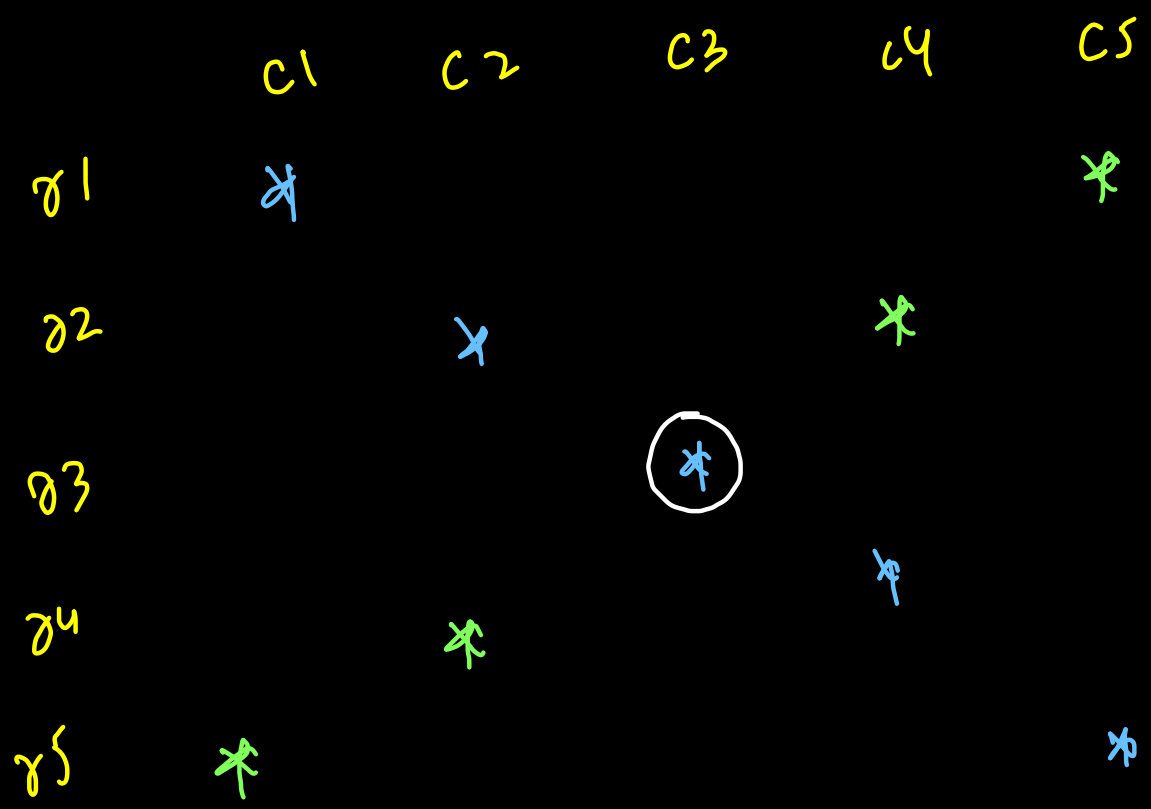
```
0 -1 -2 -3 -4
1 0 -1 -2 -3
2 1 0 -1 -2
3 2 1 0 -1
4 3 2 1 0
```

stdin

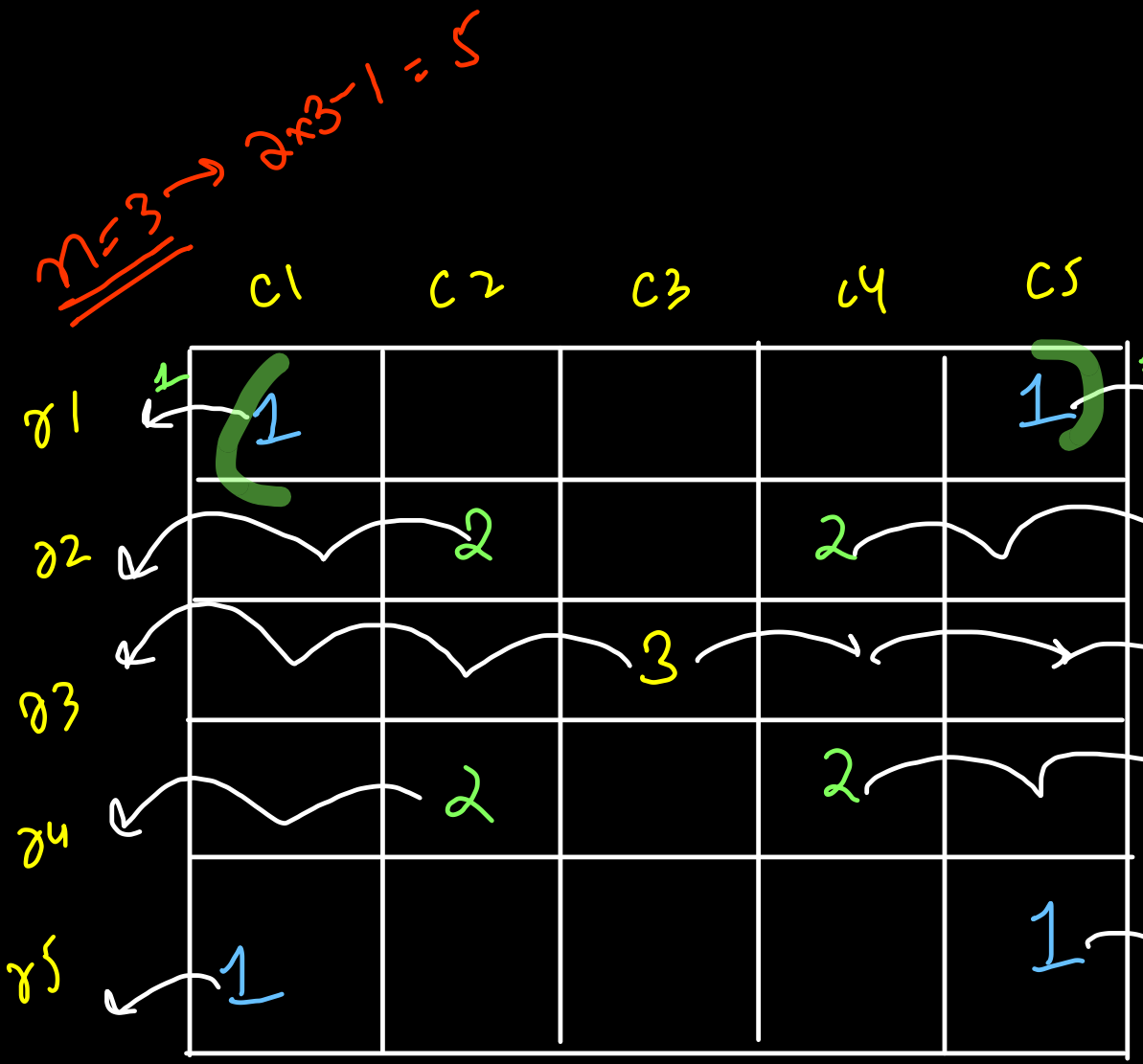
5

Starting at 10:10

X pattern or Cross Pattern



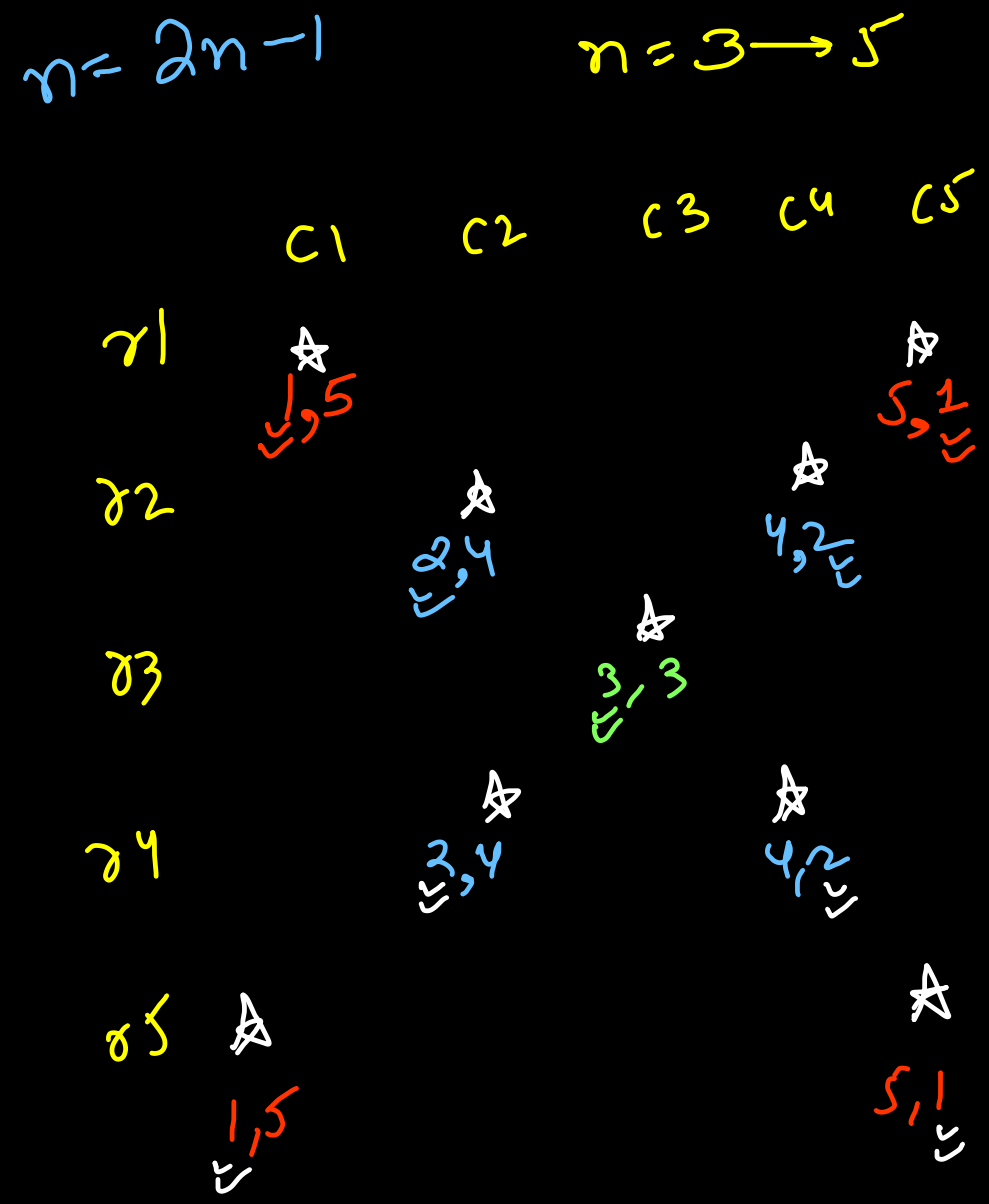

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();

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



left = col;
right = n-col+1;
} same

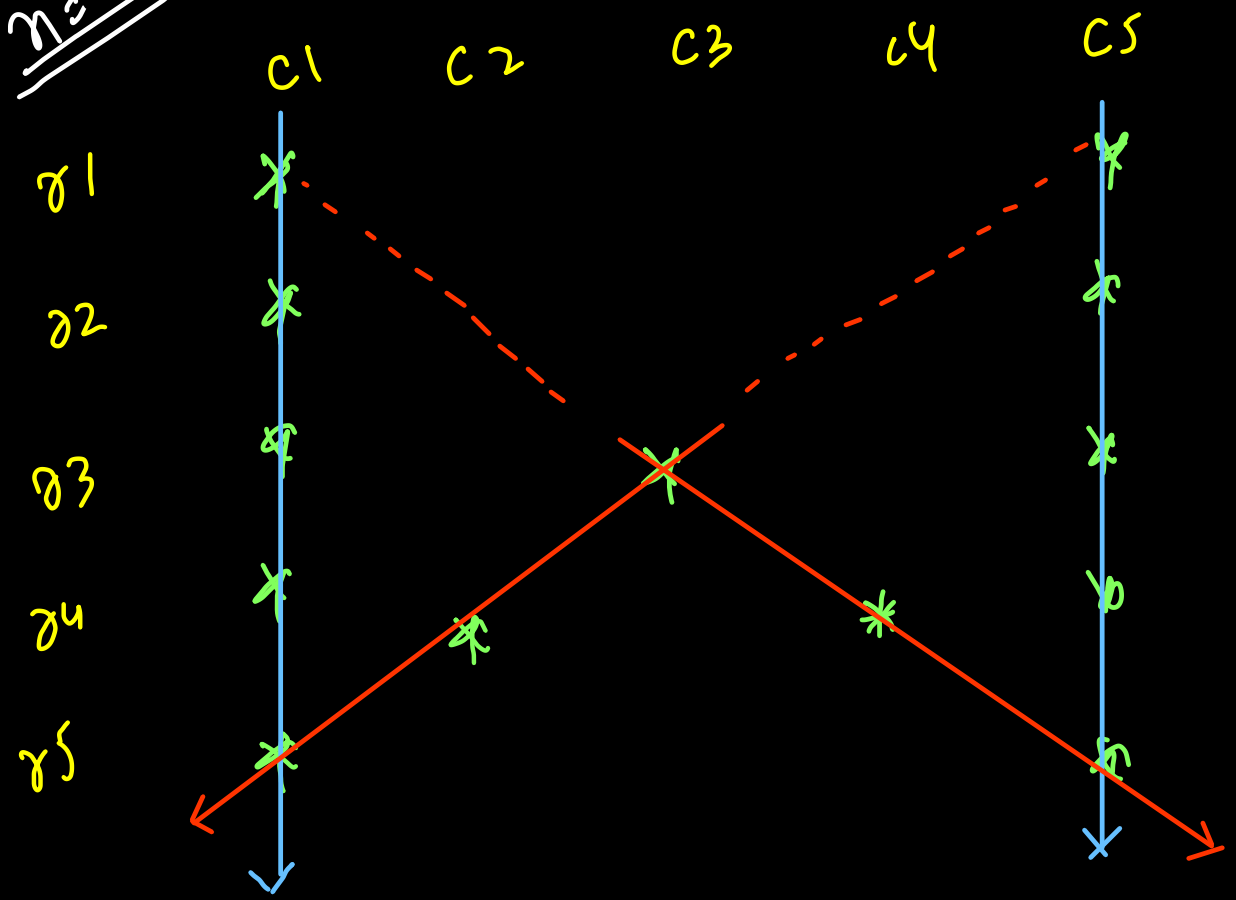
```
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();

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

```java
public static void ninjaPattern(int n) {
    n = 2 * n - 1; // rows

    for(int row = 1; row <= n; row++){
        for(int col = 1; col <= n; col++){
            if(row - col == 0 || row + col == n + 1){
                int left = col, right = n - col + 1;
                int min = Math.min(left, right);
                System.out.print(min + " ");
            } else {
                System.out.print("  ");
            }
        }
        System.out.println();
    }

}
```



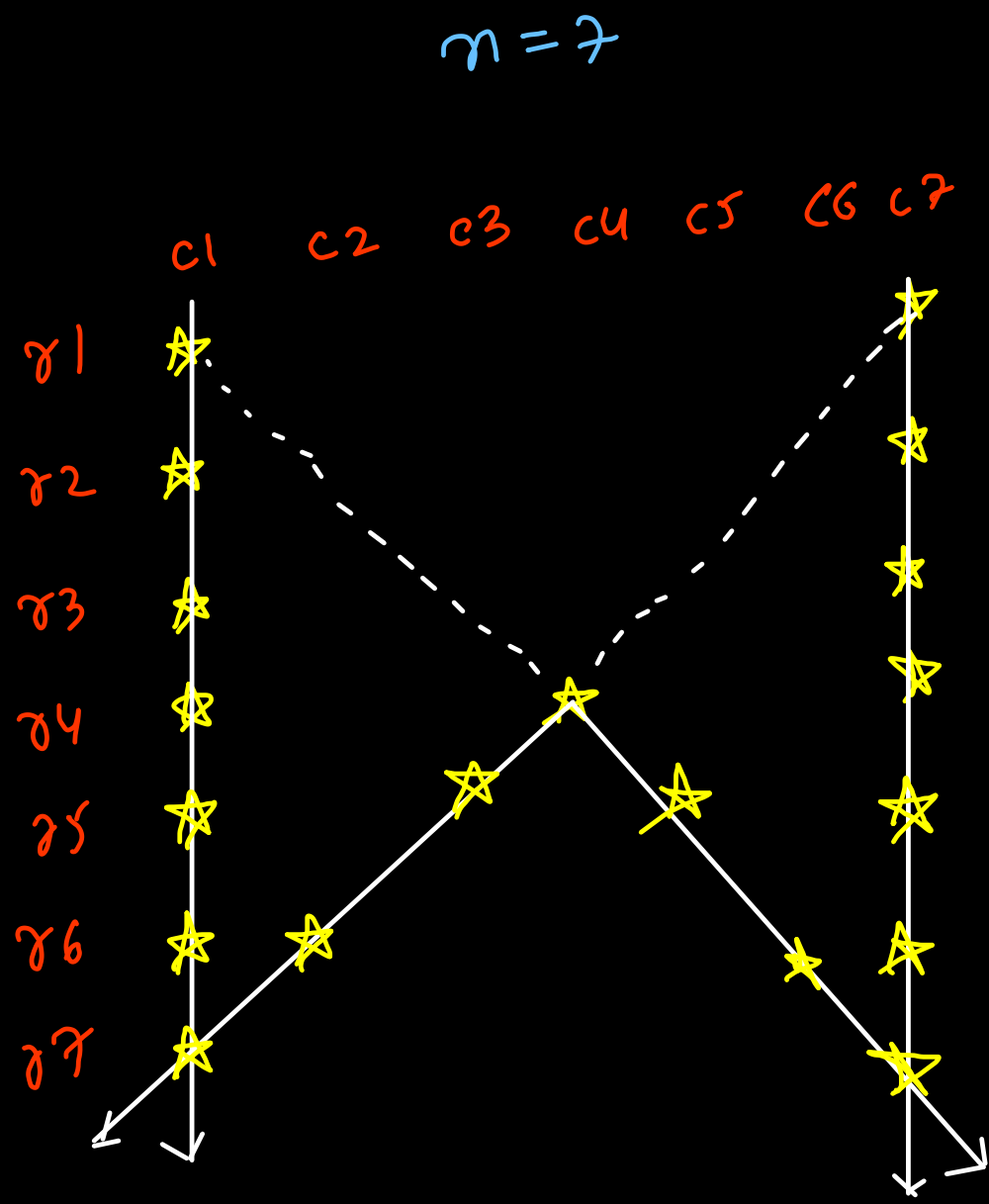
W Pattern

n = 5




```
if( col==1 || col==n )
    || (row > n/2
        && row == col)
    || (row>n/2
        && row+col==n+1))
```

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

    for(int row = 1; row <= n; row++){
        for(int col = 1; col <= n; col++){
            if(col == 1 || col == n || (row > n/2 && row == col)
                || (row > n/2 && row + col == n + 1)){
                System.out.print("*   ");
            } else {
                System.out.print("    ");
            }
        }
        System.out.println();
    }
```



Fibonacci Triangle

n=4

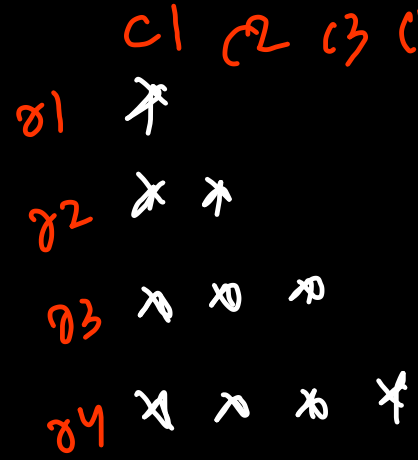
0

1 1

2 3 5

8 13 21 34

step ① Structur



int n=0

n=0 : 0

n=1 : 1

n>=2 : a, b, c

n++

$n = 4$

	c1	c2	c3	c4
r1	0			
r2	1	1		
r3	2	3	5	
r4	8	13	0	0

$a = 0$

$b = 1$

$c = 1$

Count = 0

- Count == 0 → $f_0 = 0$ (a)
- Count == 1 → $f_1 = 1$ (b)
- Count >= 2 → $c = a + b$

f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7
0	1	1	2	3	5	8	13
					a	b	c

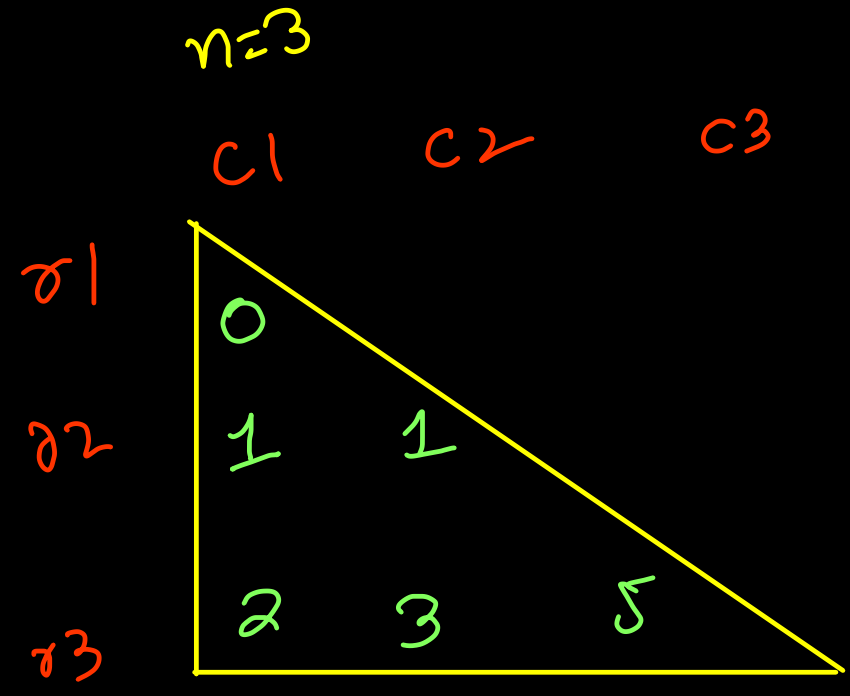
Count == 0 ~~X~~

~~r2 r3 r4 r7~~

```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 row = 1; row <= n; row++){
        for(int col = 1; col <= row; col++){
            System.out.print(a + " ");
            a = b;
            b = c;
            c = a + b;
        }
        System.out.println();
    }
}
```

f0 f1 f2

two pointer

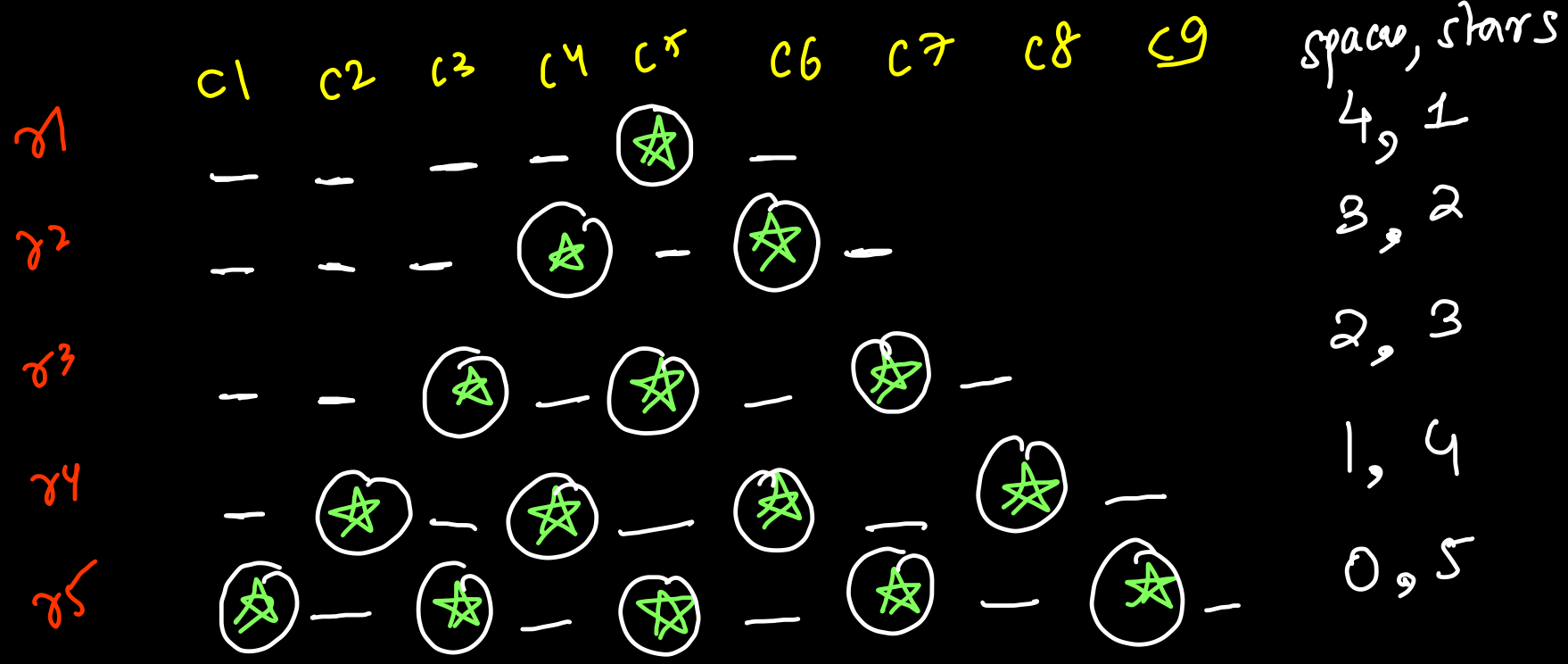


f0	f1	f2	f3	f4	f5	f6	f7	f8
0	1	1	2	3	5	8	13	21
						a	b	c

Pyramid Pattern

n = 5

Pyramid



Spaces = (n-1), stars = 1

↓ spaces--

↓ stars++

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

    int spaces = n - 1, stars = 1;
    for(int row = 1; row <= n; row++){
        for(int count = 1; count <= spaces; count++){
            System.out.print(" ");
        }
        for(int count = 1; count <= stars; count++){
            System.out.print("*");
        }
        spaces--; stars++;
        System.out.println();
    }
```

```
Finished in 129 ms
    *
   **
  ***
 ****
*****
```

Lower Right Triangle

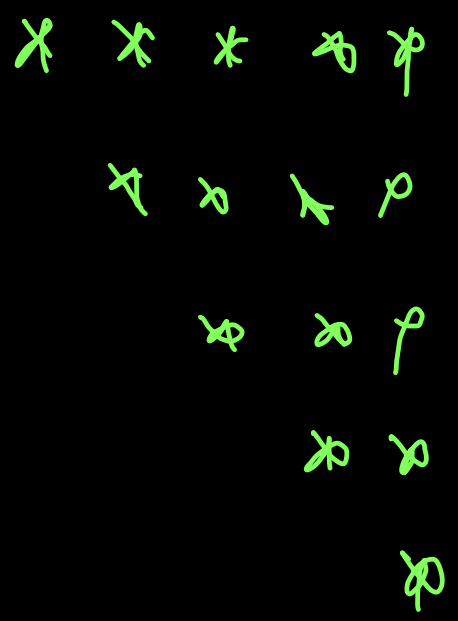
```java
int spaces = n - 1, stars = 1;
for(int row = 1; row <= n; row++){
    for(int count = 1; count <= spaces; count++){
        System.out.print(" ");
    }
    for(int count = 1; count <= stars; count++){
        System.out.print("* ");
    }
    spaces--; stars++;
    System.out.println();
}
```

add space after star

```
Finished in 157 ms
    *
   * *
  * * *
 * * * *
* * * * *
```

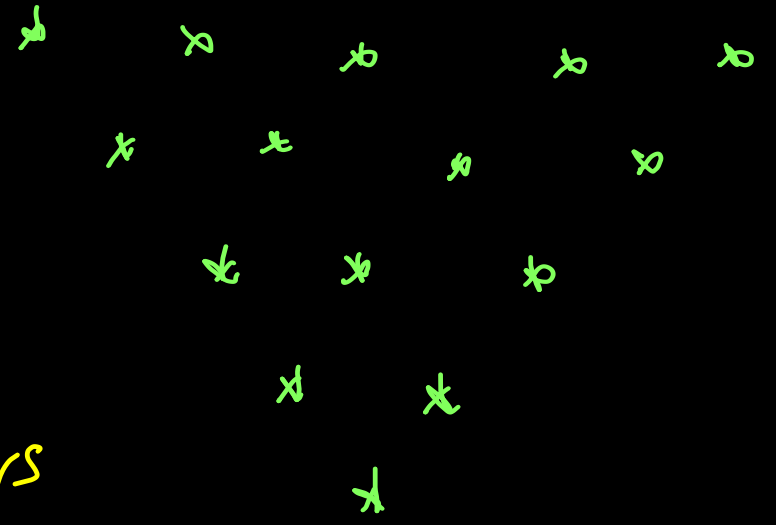
Pyramid

Upper Right Triangle



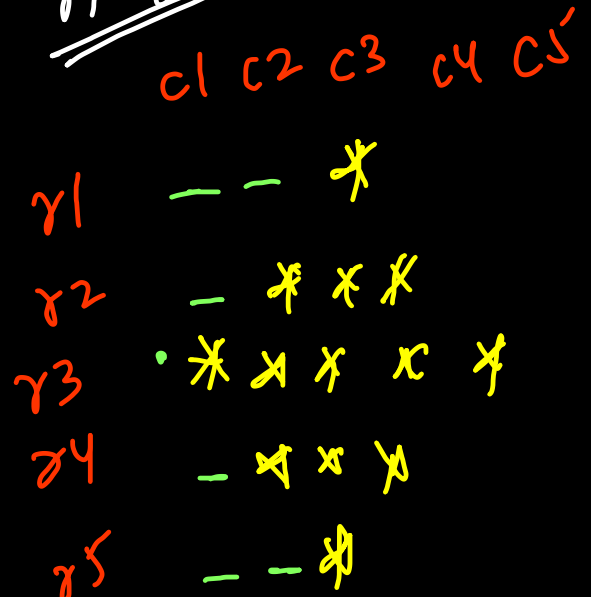
add space in b/w stars

Inverted Pyramid

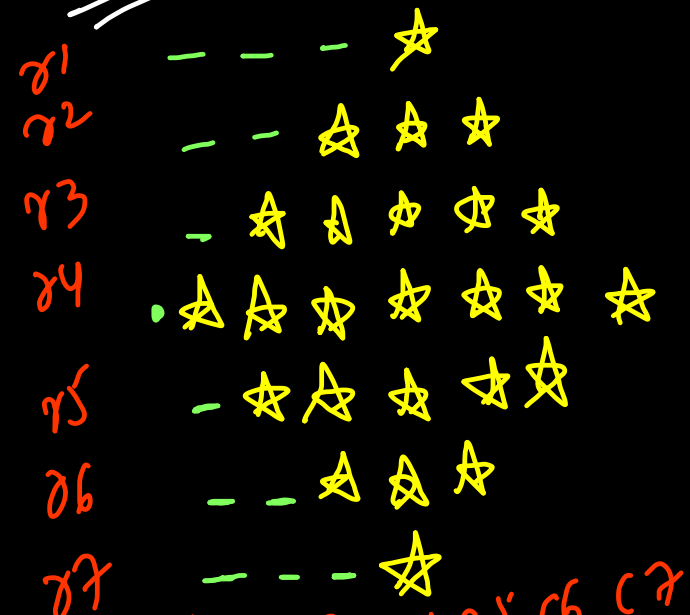


Diamond pattern

n = 5



n = 7



Stars & Spaces

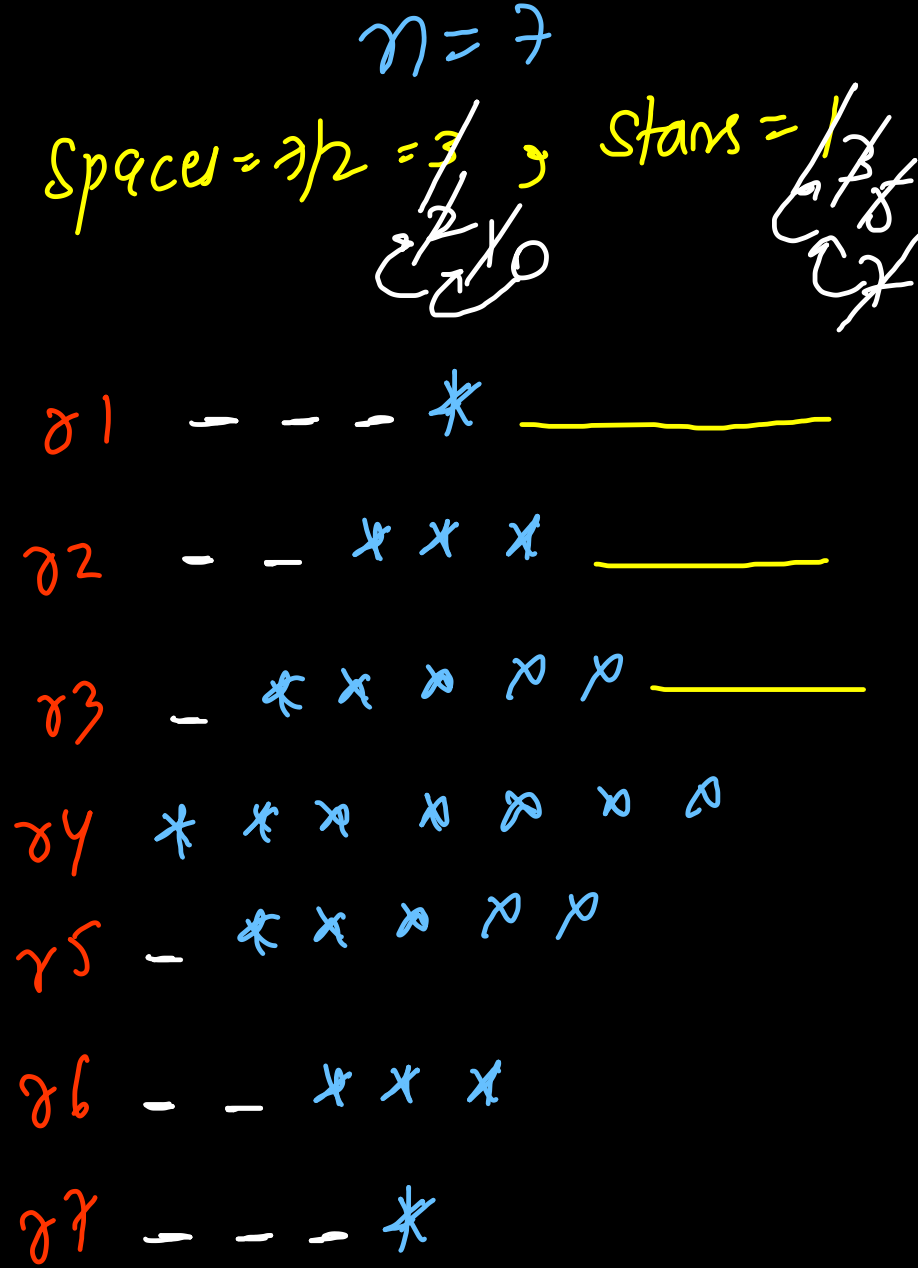
Spaces = n/2

stars = 1

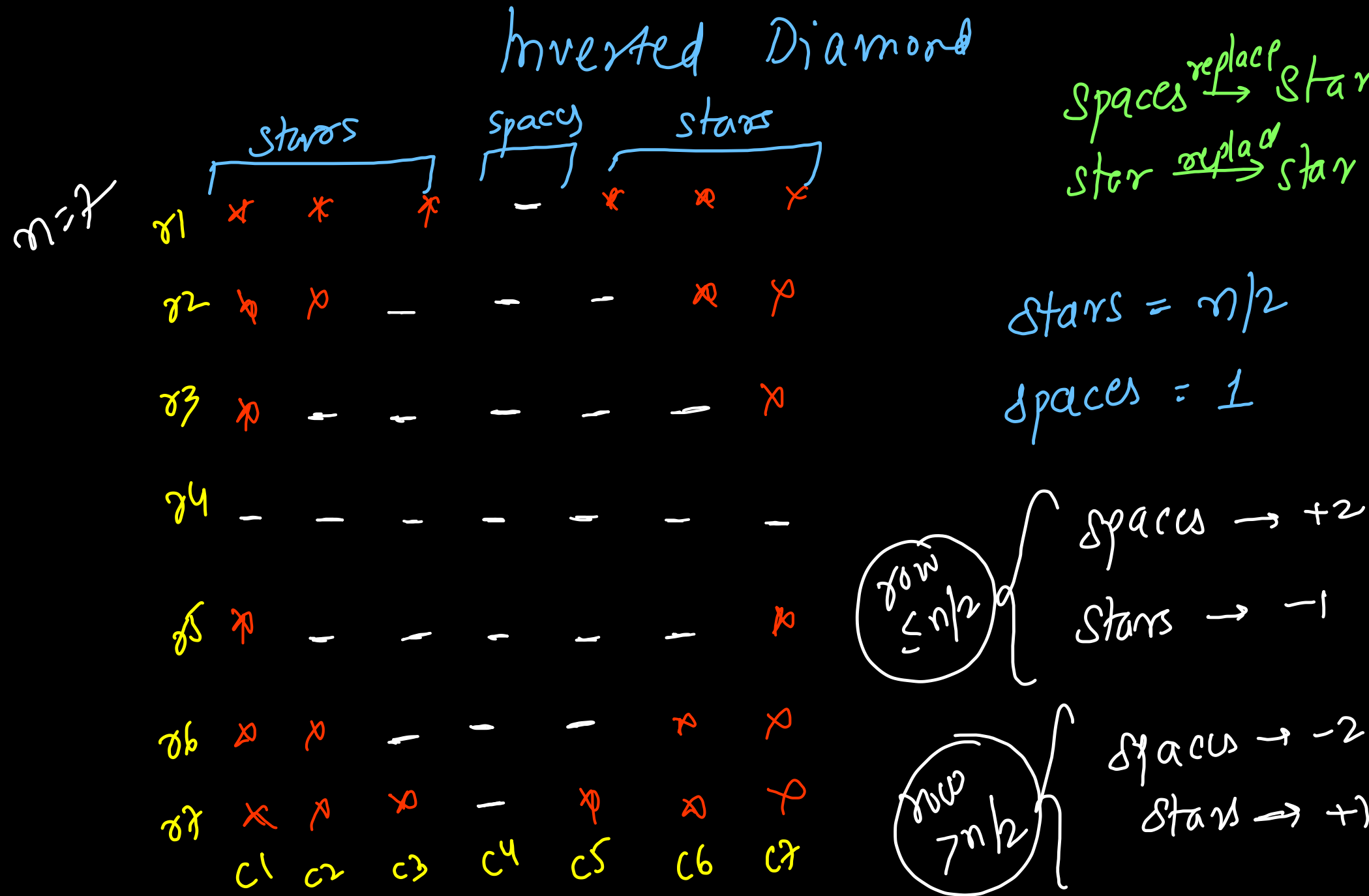
r < n/2 {
- stars → +2
- spaces → −1

r > n/2 {
- stars → −2
- spaces → +1

```java
public static void printPattern(int n) {
    int spaces = n / 2, stars = 1;

    for(int row = 1; row <= n; row++){
        for(int count = 1; count <= spaces; count++){
            System.out.print(" ");
        }
        for(int count = 1; count <= stars; count++){
            System.out.print("*");
        }
        System.out.println();

        if(row <= n/2){
            stars += 2;
            spaces--;
        } else {
            stars -= 2;
            spaces++;
        }
    }
}
```



Inverted Diamond



Spaces $\xrightarrow{\text{replace}}$ Star

Star $\xrightarrow{\text{replace}}$ Star

stars = n/2

spaces = 1

row $\leq n/2$ { spaces → +2, stars → −1

row $> n/2$ { spaces → −2, stars → +1

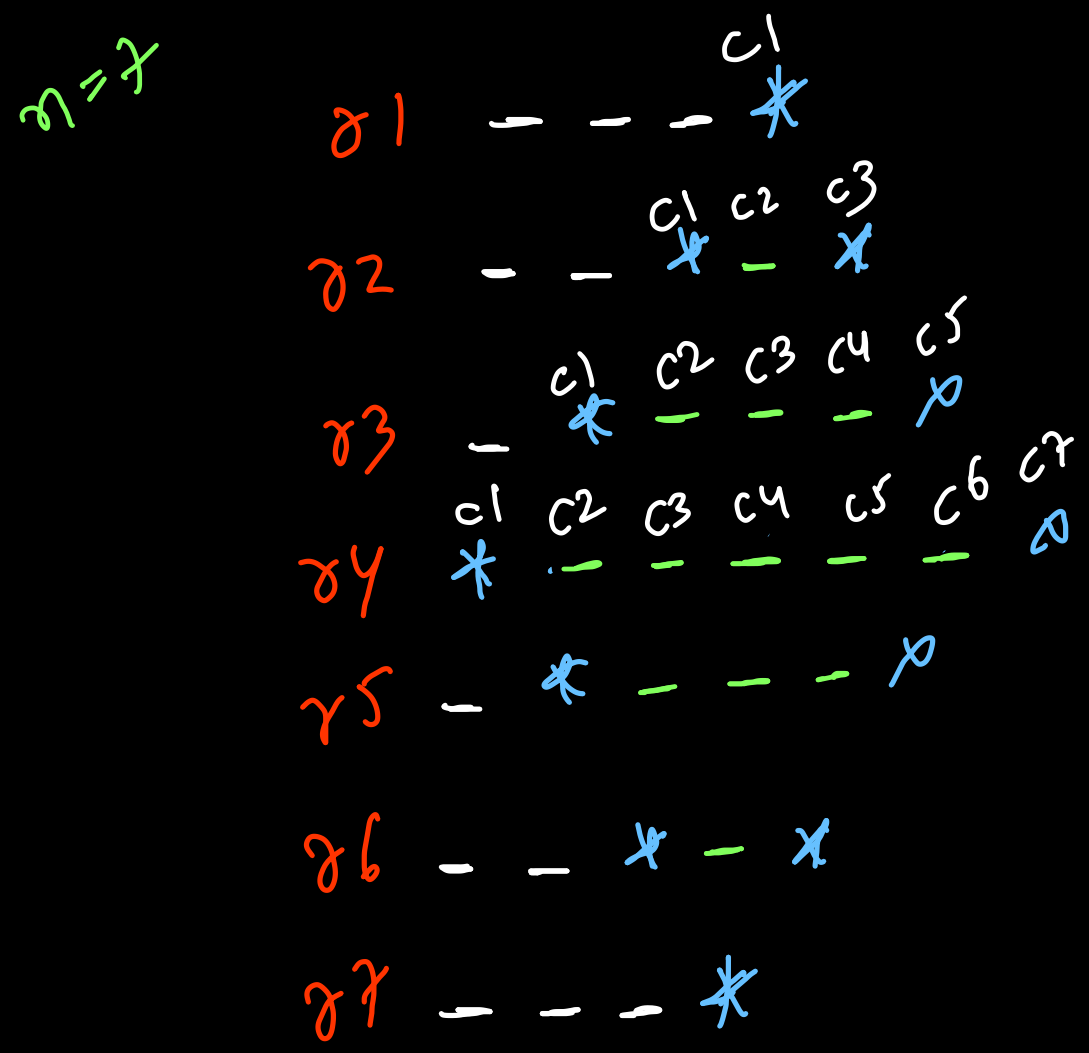
```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int stars = n / 2, spaces = 1;

    for(int row = 1; row <= n; row++){
        for(int count = 1; count <= stars; count++){
            System.out.print("*");
        }
        for(int count = 1; count <= spaces; count++){
            System.out.print(" ");
        }
        for(int count = 1; count <= stars; count++){
            System.out.print("*");
        }
        System.out.println();

        if(row <= n/2){
            spaces += 2;
            stars--;
        } else {
            spaces -= 2;
            stars++;
        }
    }
```

3 inner loops

n = 7

```
Finished in 138 ms
*** ***
**   **
*     *

*     *
**   **
*** ***
```

Diamond border




```java
public static void printPattern(int n) {
    int spaces = n / 2, stars = 1;

    for(int row = 1; row <= n; row++){
        for(int count = 1; count <= spaces; count++){
            System.out.print(" ");
        }
        for(int count = 1; count <= stars; count++){
            System.out.print("*");
        }
        System.out.println();

        if(row <= n/2){
            stars += 2;
            spaces--;
        } else {
            stars -= 2;
            spaces++;
        }
    }
}
```

if(count==1 || count==stars) Syso("*");

else Syso(" ");

```java
int spaces = n / 2, stars = 1;

for(int row = 1; row <= n; row++){
    for(int count = 1; count <= spaces; count++){
        System.out.print(" ");
    }
    for(int count = 1; count <= stars; count++){
        if(count == 1 || count == stars){
            System.out.print("*");
        } else {
            System.out.print(".");
        }
    }
    System.out.println();

    if(row <= n/2){
        stars += 2;
        spaces--;
    } else {
        stars -= 2;
        spaces++;
    }
}
```

every row's first last should be printed otherwise Space

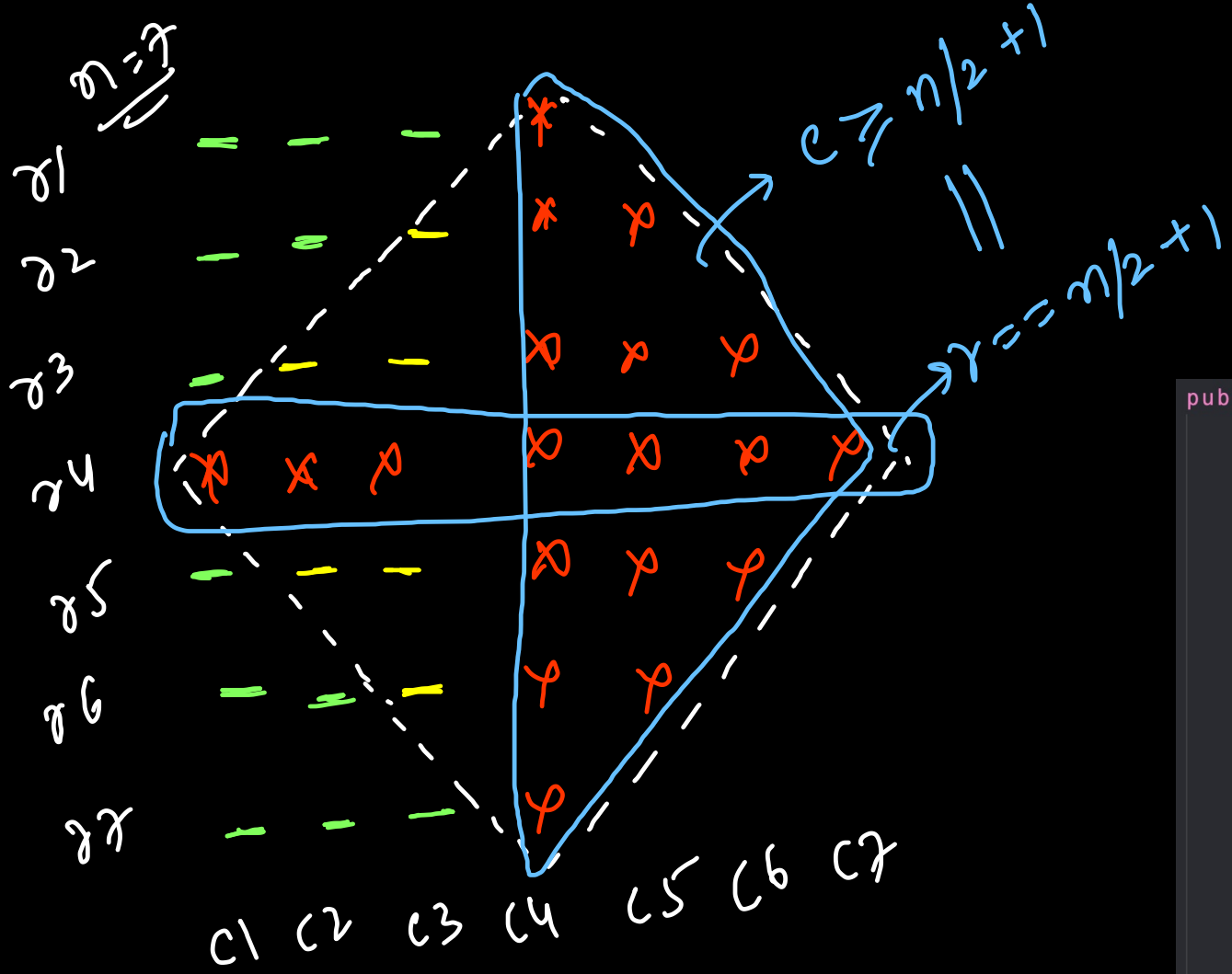
Finished in 135 ms

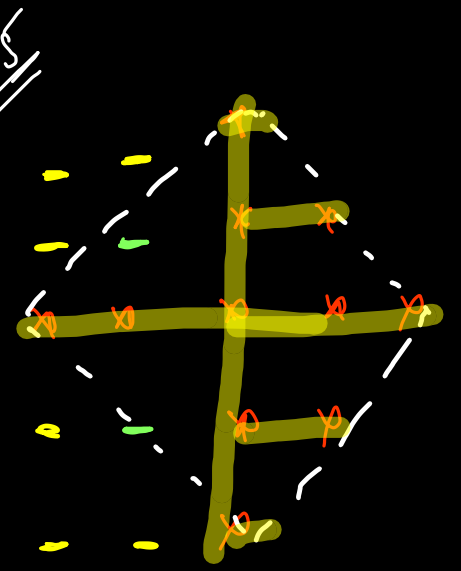
```
   *
  *.*
 *...*
*.....*
 *...*
  *.*
   *
```

stdin

7

Arrow Pattern



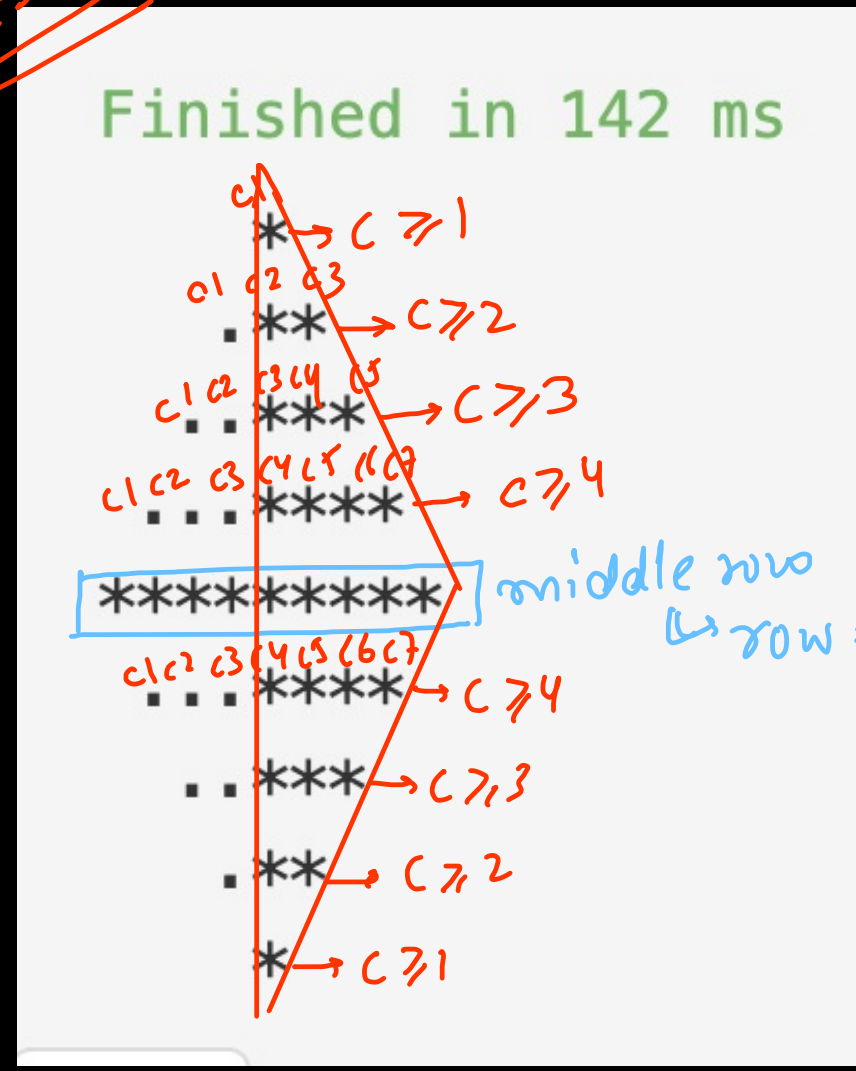



```java
public static void printPattern(int n) {
    int spaces = n / 2, stars = 1;

    for(int row = 1; row <= n; row++){
        for(int count = 1; count <= spaces; count++){
            System.out.print(" ");
        }
        for(int count = 1; count <= stars; count++){
            System.out.print("*");
        }
        System.out.println();

        if(row <= n/2){
            stars += 2;
            spaces--;
        } else {
            stars -= 2;
            spaces++;
        }
    }
}
```

if(?) Syso("x")
else Syso(" ")

```java
int spaces = n / 2, stars = 1;

for(int row = 1; row <= n; row++){
    for(int count = 1; count <= spaces; count++){
        System.out.print(" ");
    }
    for(int count = 1; count <= stars; count++){
        if(row == n/2 + 1 || count >= stars / 2 + 1){
            System.out.print("*");
        } else {
            System.out.print(".");
        }
    }
    System.out.println();

    if(row <= n/2){
        stars += 2;
        spaces--;
    } else {
        stars -= 2;
        spaces++;
    }
}
```

n=7

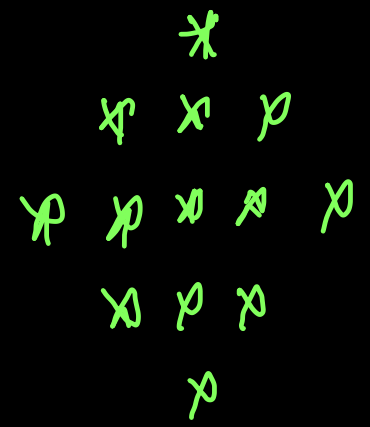


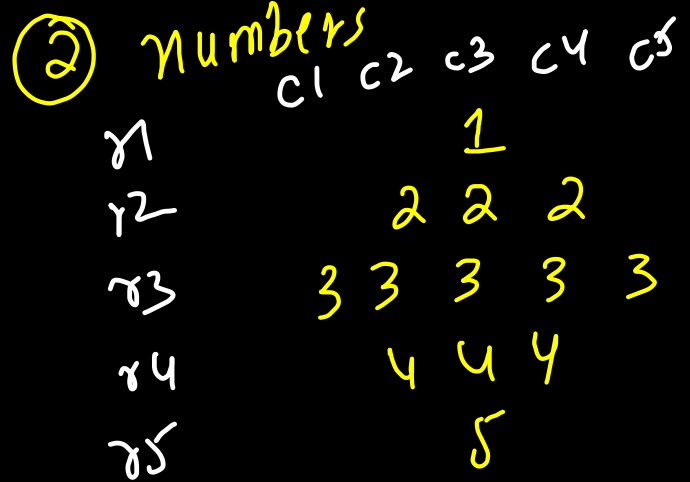
$n = 5$

	c1	c2	c3	c4	c5
r1			1		
r2		2	3	2	
r3	3	4	5	4	3
r4		2	3	2	
r5			1		

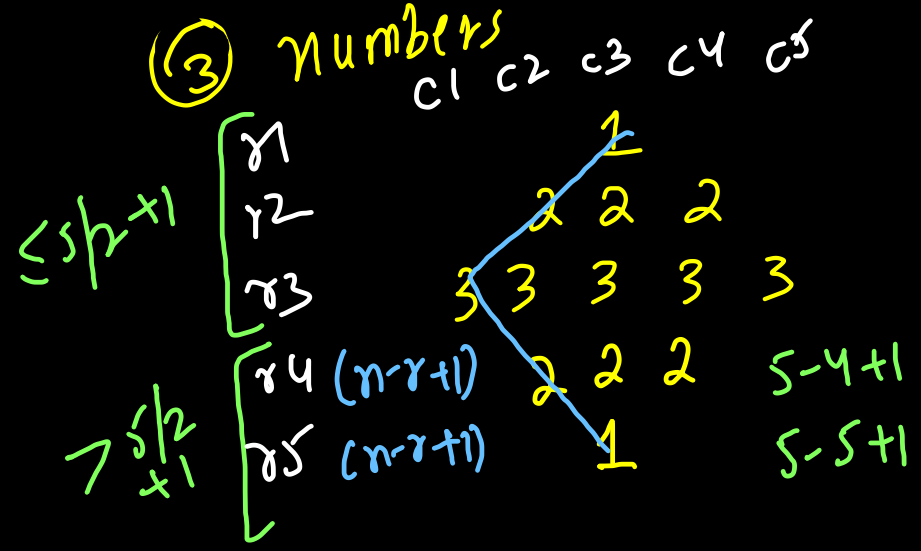
① structure (stars)

↳ diamond

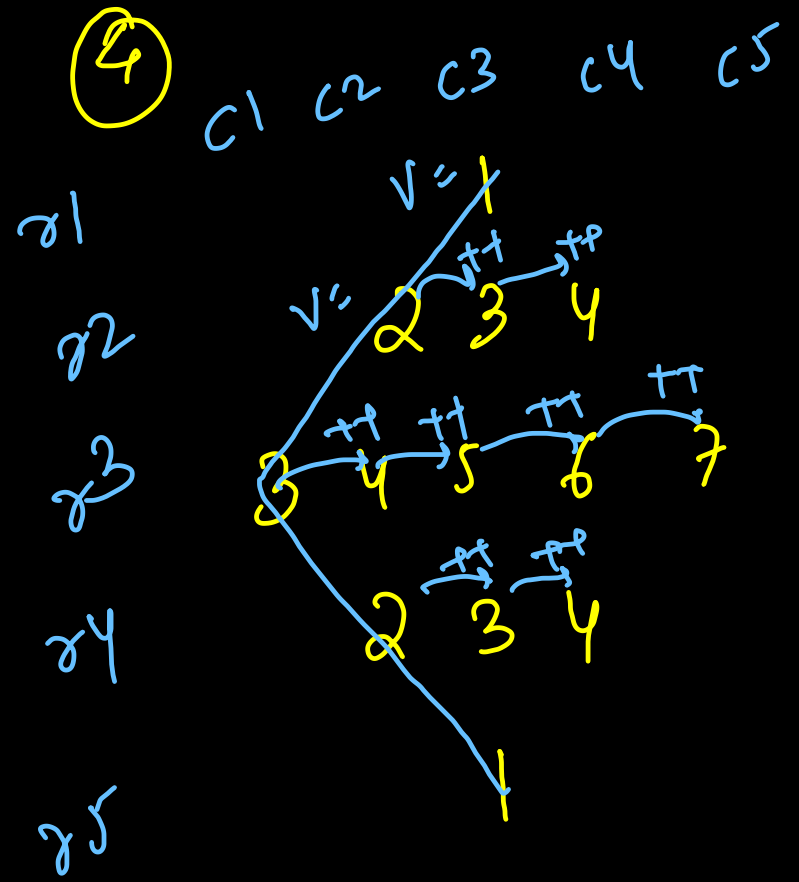




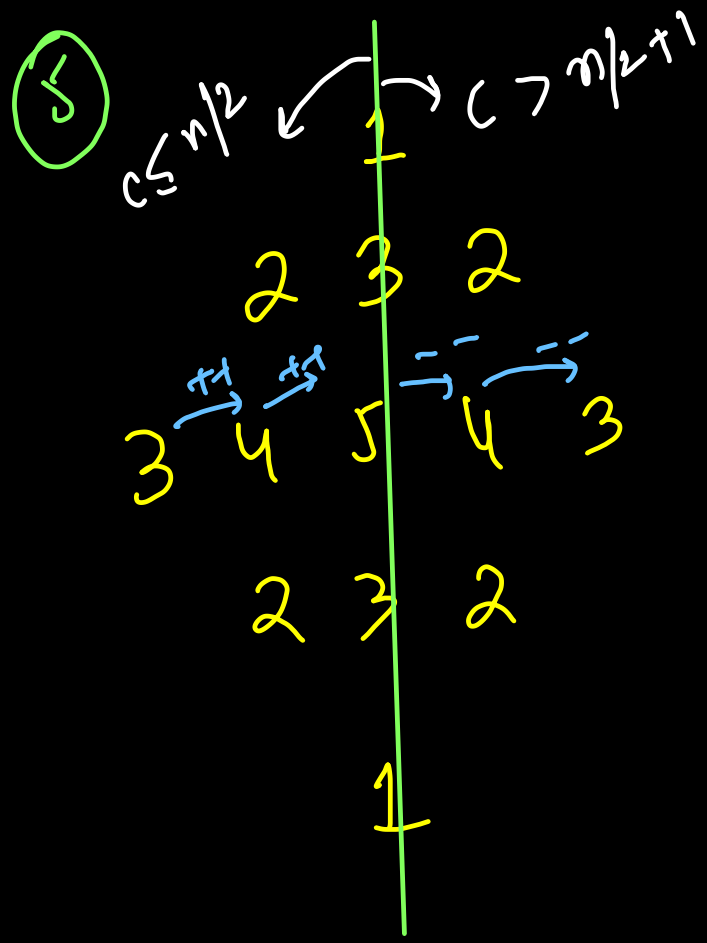

```java
int spaces = n / 2, stars = 1;
for(int row = 1; row <= n; row++){
    for(int count = 1; count <= spaces; count++){
        System.out.print(" ");
    }

    int val = row;
    for(int count = 1; count <= stars; count++){
        System.out.print(val);
    }
    System.out.println();

    if(row <= n/2){
        stars += 2;
        spaces--;
    } else {
        stars -= 2;
        spaces++;
    }
}
```




```java
int spaces = n / 2, stars = 1;
for(int row = 1; row <= n; row++){
    for(int count = 1; count <= spaces; count++){
        System.out.print(" ");
    }

    int val = row;
    if(row > n/2 + 1) {
        val = n - row + 1;
    }

    for(int count = 1; count <= stars; count++){
        System.out.print(val);
    }
    System.out.println();

    if(row <= n/2){
        stars += 2;
        spaces--;
    } else {
        stars -= 2;
        spaces++;
    }
}
```




```java
int spaces = n / 2, stars = 1;
for(int row = 1; row <= n; row++){
    for(int count = 1; count <= spaces; count++){
        System.out.print(" ");
    }

    int val = row;
    if(row > n/2 + 1) {
        val = n - row + 1;
    }

    for(int count = 1; count <= stars; count++){
        System.out.print(val);
        val++;
    }
    System.out.println();

    if(row <= n/2){
        stars += 2;
        spaces--;
    } else {
        stars -= 2;
        spaces++;
    }
}
```




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

int spaces = n / 2, stars = 1;
for(int row = 1; row <= n; row++){
    for(int count = 1; count <= spaces; count++){
        System.out.print(" ");
    }

    int val = row;
    if(row > n/2 + 1) val = n - row + 1;

    for(int count = 1; count <= stars; count++){
        System.out.print(val);

        if(count <= stars/2) val++;
        else val--;
    }
    System.out.println();

    if(row <= n/2){
        stars += 2;
        spaces--;
    } else {
        stars -= 2;
        spaces++;
    }
}
```

Variable convention, func^n name

↳ camelcase matching

sum, product, answer

sumOfMarks, dbConnection

Classname

↳

Solution, System, String, Scanner, StudentUserInterface

Constants

↳ snake-case

Integer.MAX_VALUE

Integer.MIN_VALUE

Ternary Operator → Conditional statement

variable = (condition) ? value1 : value2 ;

true false

```
String result = (n%2==0) ? "even" : "odd";
```

syntax

```
String result = " ";
```

5 → "odd"

6 → "even"

⟹

```
if (n % 2 == 0) result = "even";
else result = "odd";
```

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

    String result = (n % 2 == 0) ? "even" : "odd";
    System.out.println(result);
```

n = 7 → "odd"

n = 6 → "even"

maximum of a & b

```
int c = (a > b) ? a : b;
```

```java
int max = (a > b) ? a : b;
int min = (a < b) ? a : b;
System.out.println(max + " " + min);
```

minimum of a & b

```
int c = (a < b) ? a : b;
```

Switch case

Calculator → Basic (+, -, /, *)

int int character
a , b , operator

5 + 2

3 / 7

7 x 6

5 - 8

10 # 2

```
while(true){
    int a = scn.nextInt();
    int b= scn.nextInt();
    char op = scn.next().charAt(0);
    switch(op){
        case '+' : {
        }
        case '-' : {
        }
        case '/' : {
        }
        case '*' : {
        }
        default : {
        }
    }
}
```

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

while(true){
    int a = scn.nextInt();
    int b = scn.nextInt();
    char operator = scn.next().charAt(0);

    switch(operator){
        case '+': {
            System.out.println(a + b);
            break;
        }

        case '-': {
            System.out.println(a - b);
            break;
        }

        case '/': {
            System.out.println(a / b);
            break;
        }

        case '*': {
            System.out.println(a * b);
            break;
        }

        default: {
            System.out.println("Invalid Input");
            return;
        }
    }
```

```
Finished in 121 ms
3
7
10
Invalid Input
```

stdin

```
5 2 -
5 2 +
5 2 *
5 4 #
```

Methods (or) Functions

eg: main()

print(), println(), ...

nextInt(), next(), ...

max(), min(), pow(),

need (why)
↳ clean code

what
↳ submodule of code

how?

How?

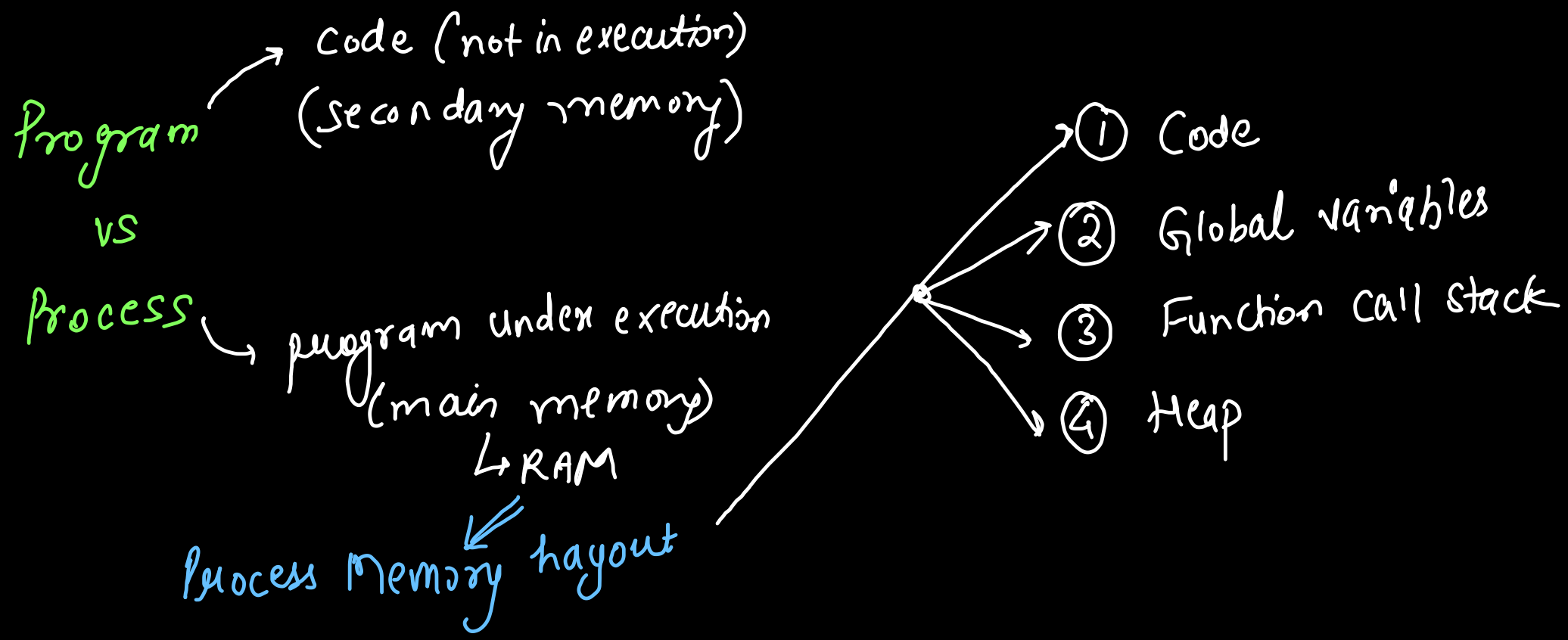
accessmodifier static returntype functionName (arguments) {
or parameter

// write your fn body here

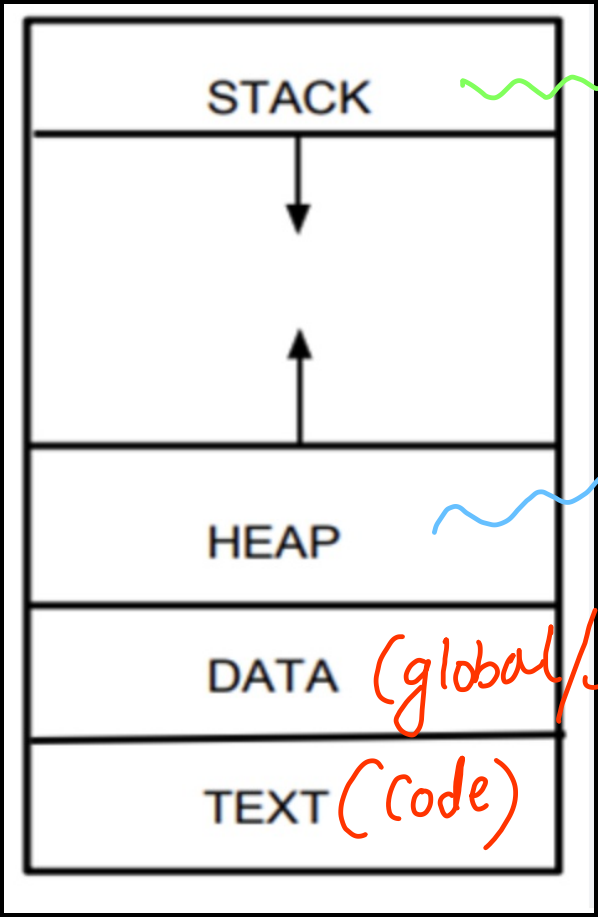
}

eg public static void printTime ( ) {
nothing

}


Program
Code (not in execution)
(secondary memory)
vs
Process
program under execution
(main memory)
↳ RAM
Process Memory layout
1 Code
2 Global variables
3 Function call stack
4 Heap

Process Memory Layout



STACK → function call stack * → primitive variables (&) → Reference variables
(organized form)

HEAP → objects, array, String literals, etc
(not in function scope)
(unorganized form)

DATA (global/static)

TEXT (code)

① Fn without parameter, w/o return type

```
public class Main {
    public static void printTime(){
     2.1 System.out.println("Print Time Started");
     2.2 System.currentTimeMillis();
     2.3 System.out.println("Print Time Ended");
    }

    // must, necessary
    public static void main(String[] args){
     1.1 System.out.println("Main Method Started");
     1.2 System.out.println("Main Method Ended");
    }
}
```

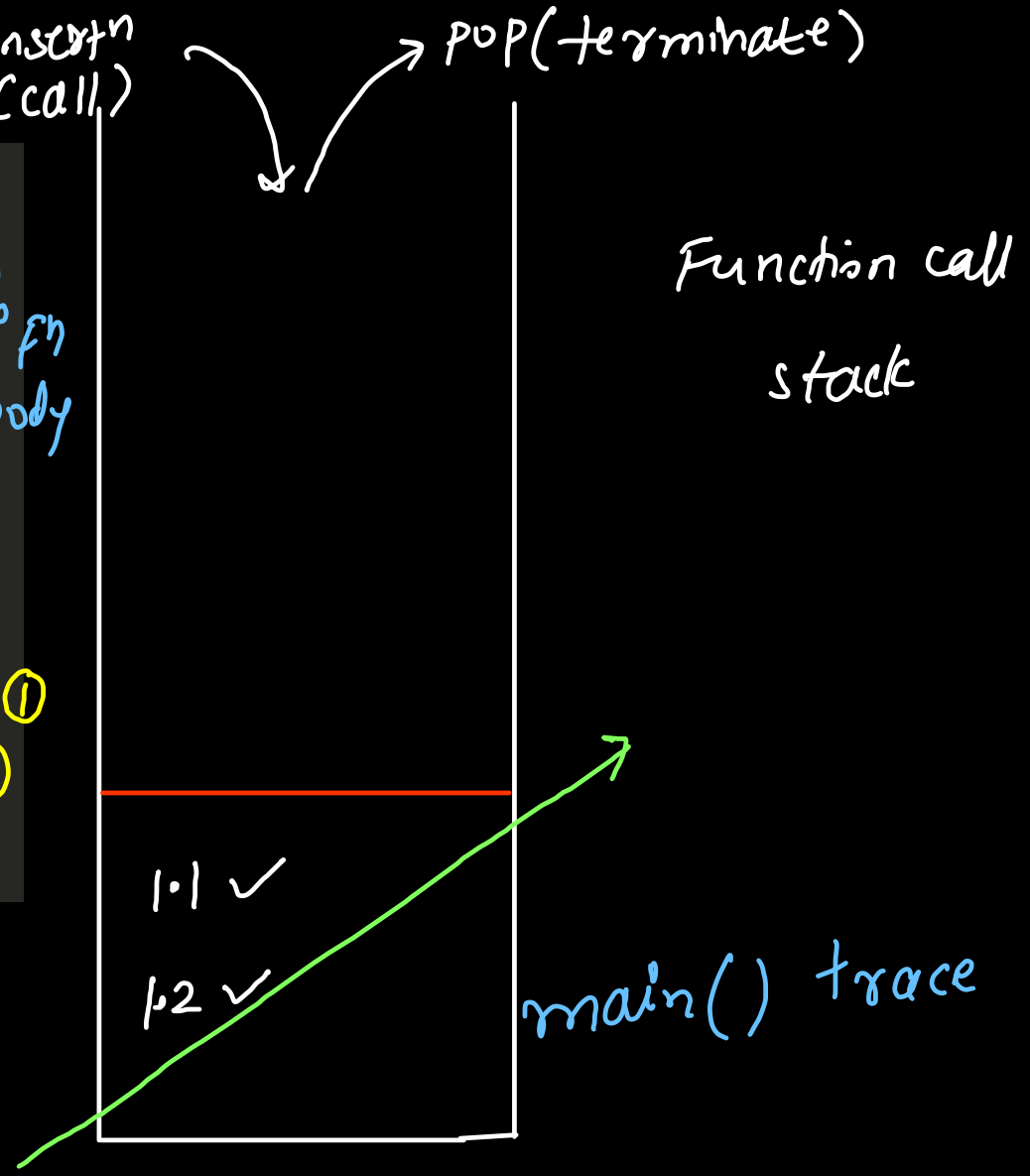
Function definition

Fn body

insertn (call)

pop (terminate)

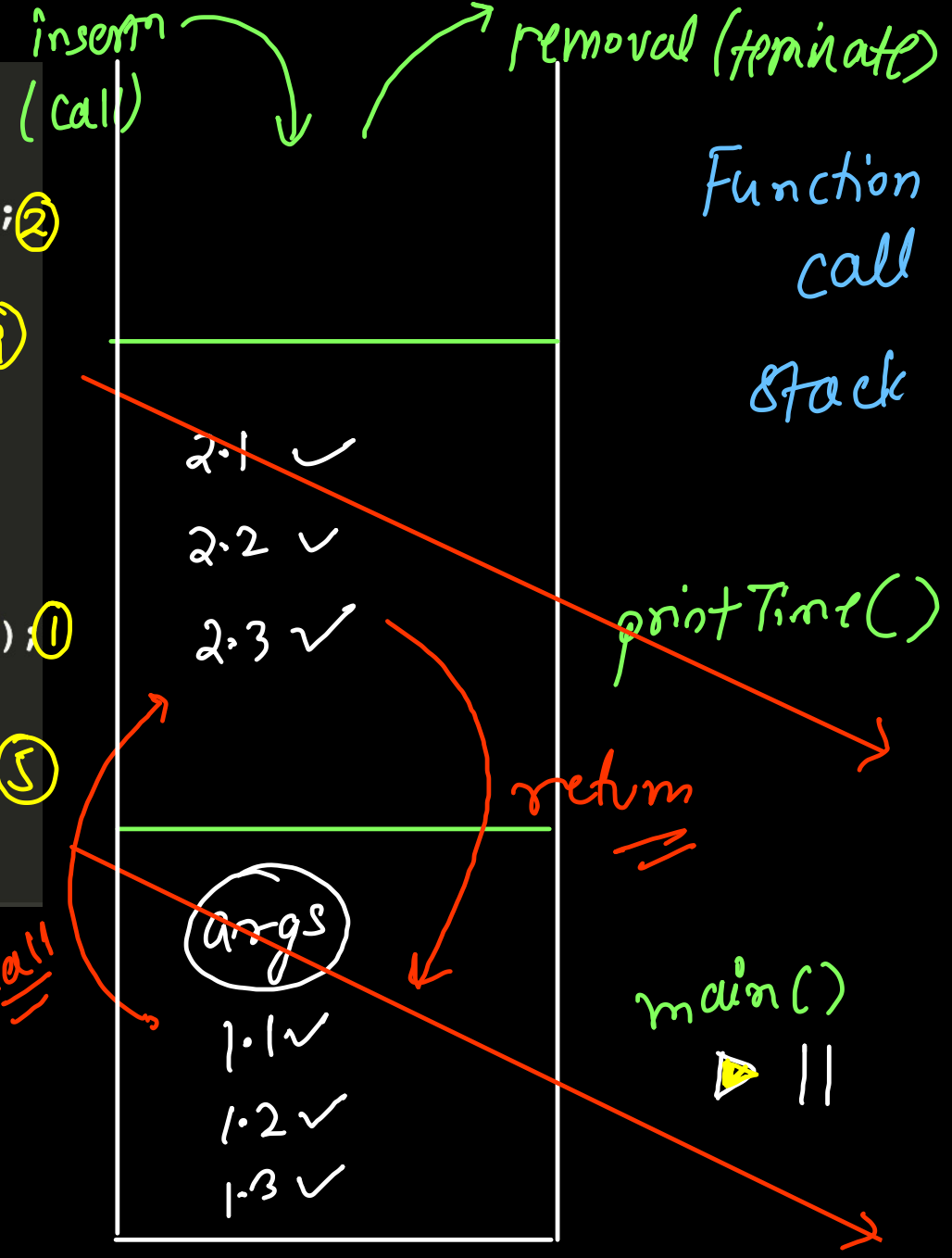
Function call stack



- main fn is the first fn to execute.

JVM

```java
public class Main {
    public static void printTime(){
        System.out.println("Print Time Started");
        System.currentTimeMillis();
        System.out.println("Print Time Ended");
    }

    // must, necessary
    public static void main(String[] args){
        System.out.println("Main Method Started");
        printTime(); // function call
        System.out.println("Main Method Ended");
    }
}
```



② Fn w/o return type, with parameters

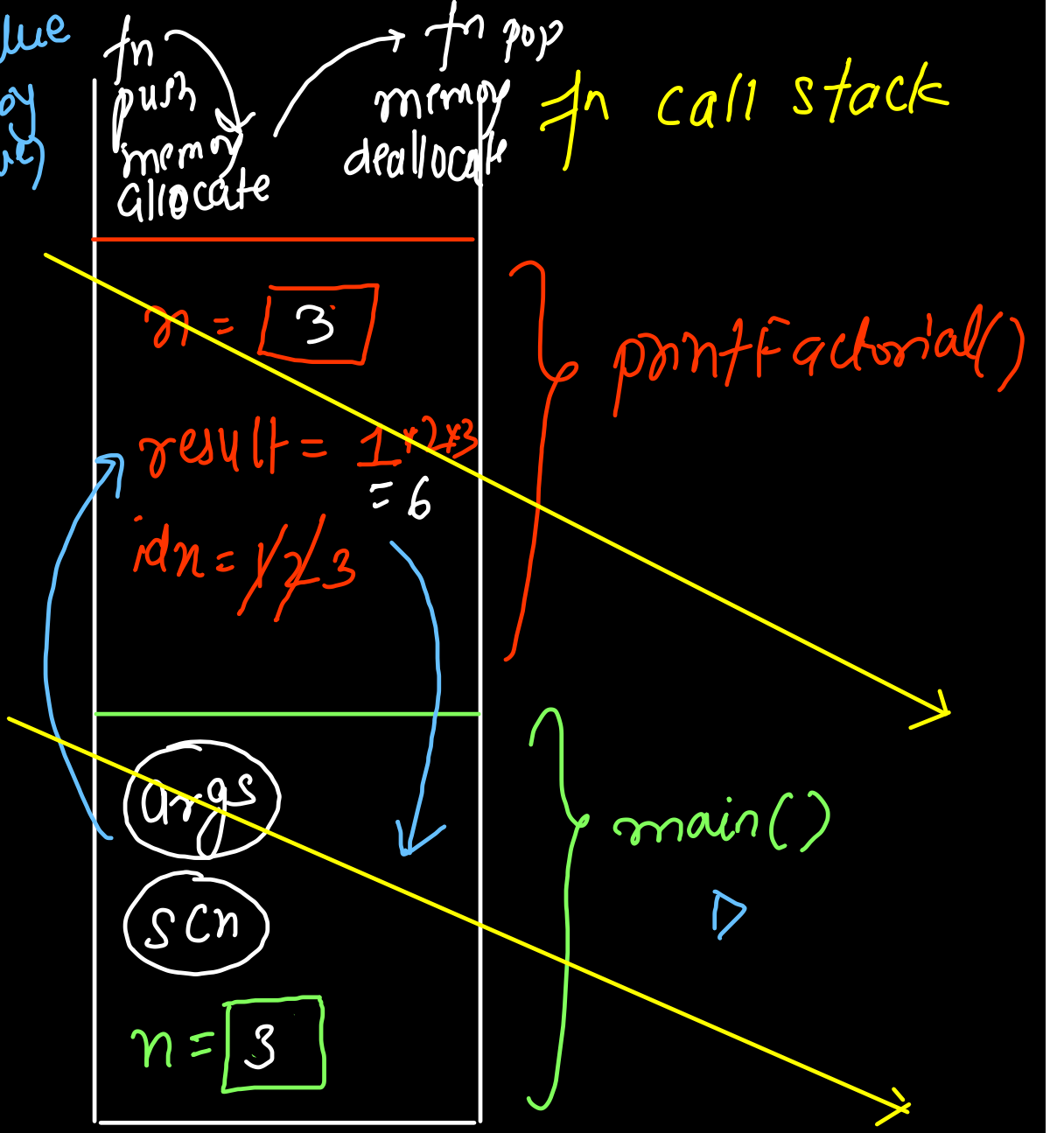
Pass by value (copy by value)

```java
public static void printFactorial(int n){
    int result = 1;
    for(int idx = 1; idx <= n; idx++){
        result = result * idx;
    }
    System.out.println("Factorial : " + result);
}

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

    int n = scn.nextInt();
    System.out.println("N = " + n);

    printFactorial(n); // function call
    System.out.println("Main Method Ended");
}
```

② Factorial = 6

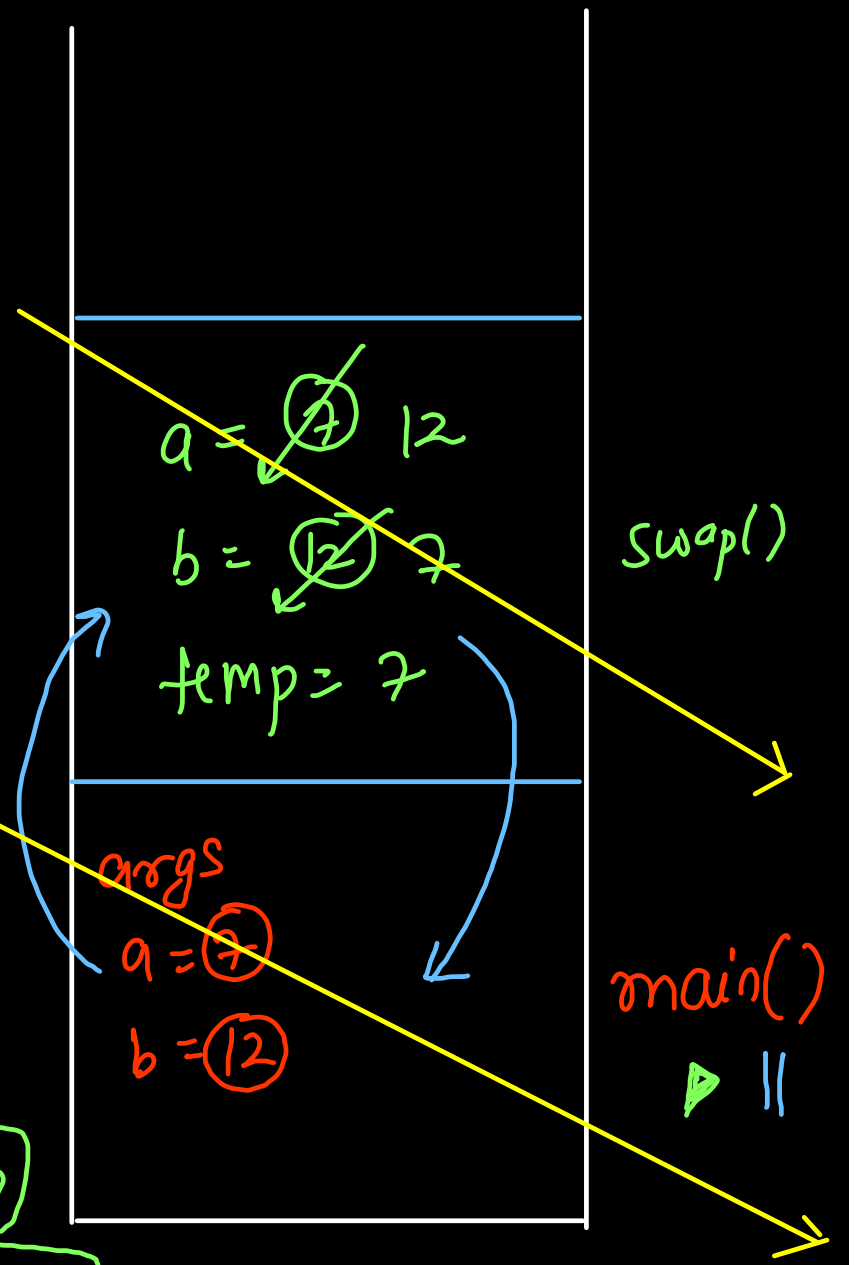
N = 3 ①

③



local fn variable
↳ lifetime Scope → fn in the Stack

```java
public static void swap(int a, int b){
    System.out.println("Inside Swap: Before Swap: " + a + " " + b);
    int temp = a;
    a = b;
    b = temp;
    System.out.println("Inside Swap: After Swap: " + a + " " + b);
}

public static void main(String[] args){
    int a = 7, b = 12;
    System.out.println("Inside Main: Before Swap: " + a + " " + b);
    swap(a, b);
    System.out.println("Inside Main: After Swap: " + a + " " + b);
}
```



① fn scope/lifetime

② fn pass by value

local variable changes doesn't persist after fn pops out.

Soln → a & b return

→ a & b as array pass

③ Fn with parameter with return type

binomial coeff

$$nC_r = \frac{n!}{(n-r)! * r!}$$

$$5C_2 = \frac{5!}{3! * 2!}$$

permutation coeff

$$nP_r = \frac{n!}{(n-r)!}$$

```java
public static int getFactorial(int x){
    int fact = 1;
    for(int idx = 1; idx <= x; idx++){
        fact *= idx;
    }
    return fact;
}

public static void main(String[] args){
    int n = 5, r = 2;

    int nfact = getFactorial(n);
    System.out.println(n + "! = " + nfact);

    int rfact = getFactorial(r);
    System.out.println(r + "! = " + rfact);

    int nmrfact = getFactorial(n - r);
    System.out.println((n - r) + "! = " + nmrfact);

    int result = nfact / (rfact * nmrfact);
    System.out.println("nCr = " + result);
}
```

120
2
6
5! = 120
2! = 2
3! = 6
nCr = 5