About Me

Incoming SDE Intern + SDE (FTE) @ **Delhivery**

SDE Intern + SDE (FTE) offer @ **Chaayos**

Technical Content Writer/Engineer @ **Scaler**

Technical Content Writer/Engineer @ **PrepBytes**

Ex – Technical Content Writer @ **InterviewBit**

Ex – Technical Content Engineer @ **Pepcoding**

Offers from other Ed-tech institutions like Unstop (Dare2Compete), TuteDude, etc.

Syllabus

Time & Space Complexity

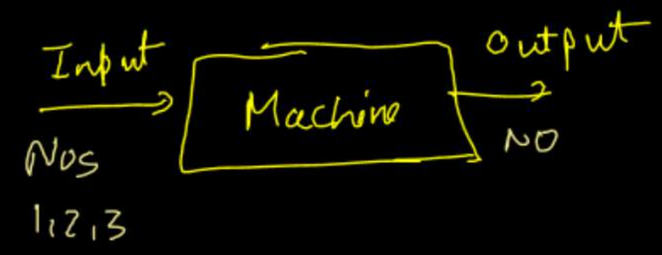
1. Getting Started
2. Arrays & Strings
   - → Two Pointer + Sliding Window
   - → Greedy Algorithms
   - → Searching & Sorting
3. Recursion & Backtracking
4. Object Oriented Programming (Java)

Data Structures & Algorithms

5. Linked List
6. Stack & Queue
7. Binary Tree & Binary Search Tree
8. Hashmap & Heap
9. Graphs
10. Dynamic Programming
11. Number Theory & Bit Manipulation
12. Trie

Computer → Human convinience

↳ Machine

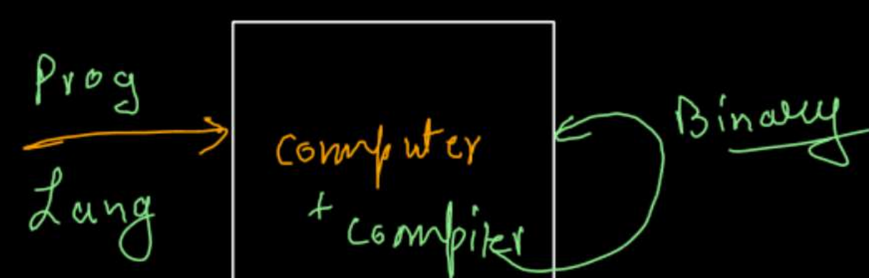


Communication

Computer understands Binary lang (0 and 1s)

Humans understand Human Lang → instructions + emotions

Prog Lang



① Compiler → Software that converts Prog Lang to Binary Lang

Input/output

```java
// "static void main" must be defined in a public class.
public class Main {
    public static void main(String[] args) {

    }
}
```

comments

```java
import java.util.*;
public class Main {
    public static void main(String[] args) {
        //this is a comment
    }
    //this is also a comment
}
```

Print in Java

```java
import java.util.*;
public class Main {
    public static void main(String[] args) {
        System.out.print("DSA classes");
    }
}
```

```java
import java.util.*;
public class Main {
    public static void main(String[] args) {
        System.out.print("Guneet Malhotra");
        System.out.print("DSA Classes");
    }
}
```

```
Finished in 69 ms
Guneet MalhotraDSA Classes
```

```java
import java.util.*;
public class Main {
    public static void main(String[] args) {
        System.out.print("Guneet Malhotra ");
        System.out.print("DSA Classes");
    }
}
```

```
Finished in 82 ms
Guneet Malhotra DSA Classes
```

```java
import java.util.*;
public class Main {
    public static void main(String[] args) {
        System.out.print("Guneet Malhotra");
        System.out.print(" DSA Classes");
    }
}
```

```
Finished in 84 ms
Guneet Malhotra DSA Classes
```

System.out.print("Rohan");

RI

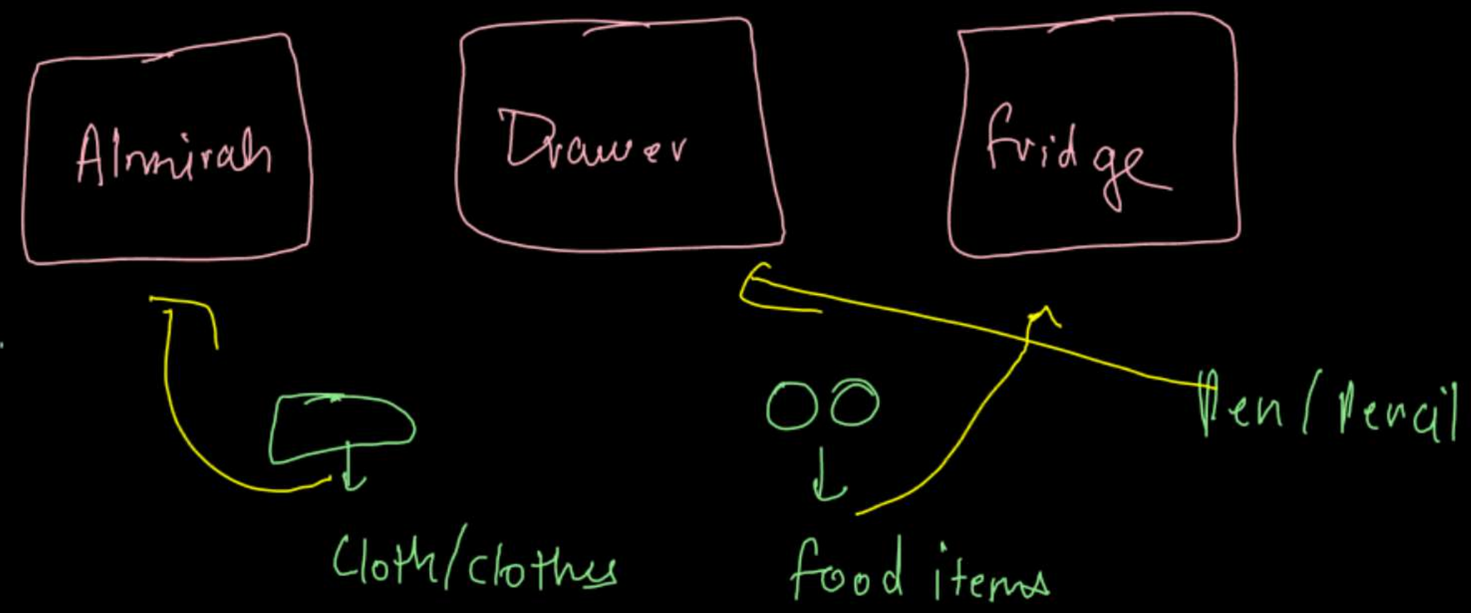
Rohan
I

```java
import java.util.*;
public class Main {
    public static void main(String[] args) {
        System.out.println("Guneet Malhotra");
        System.out.print("DSA Classes");
    }
}
```

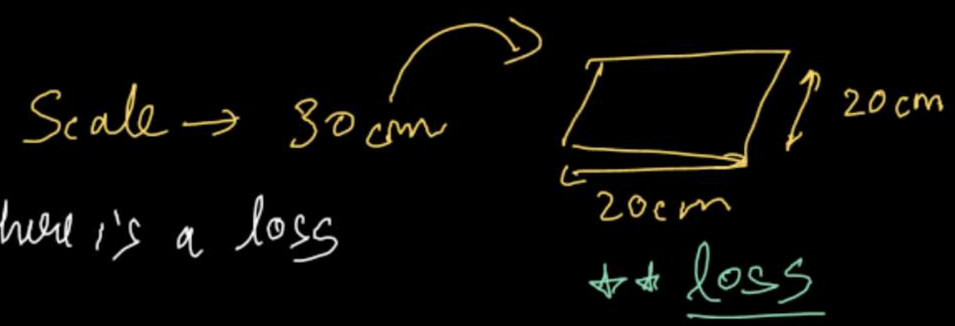
```
Finished in 122 ms
Guneet Malhotra
DSA Classes
```

Data types in Java

Type of container.



If we try to put item in small size container, there is a loss



		Size in bytes	
Integral (w/o decimal)	byte	1	30, 50, 700 etc.
	short	2	
	int	4	
	long	8	
floating pt type (with decimal)	float	4	3.141, 8.632 etc
	double	8	
Characters →	char	2 → A-Z, a-z, 0-9, #, *, etc.	
True/false →	boolean	Not fixed	

Range of Datatype

1 byte = 8 bits

$[-2^{N-1}$ to $2^{N-1}-1]$ → Here N is the no of bits.

Range of Short

Size = 2bytes = 2x8 = 16bits

$$\left[-2^{16-1} \text{ to } 2^{16-1} - 1 \right]$$

$$\underbrace{0\ 0\ 0 \cdots 0}_{16\text{ bits}}$$

$$\left[-32768 \text{ to } 32767 \right]$$

32768 → 17bit

Data loss

```java
import java.util.*;
public class Main {
    public static void main(String[] args) {
        int x = 100;
        System.out.println(x);
    }
}
```

```
Finished in 62 ms
100
```

0000 Binary Repr...

32

variable

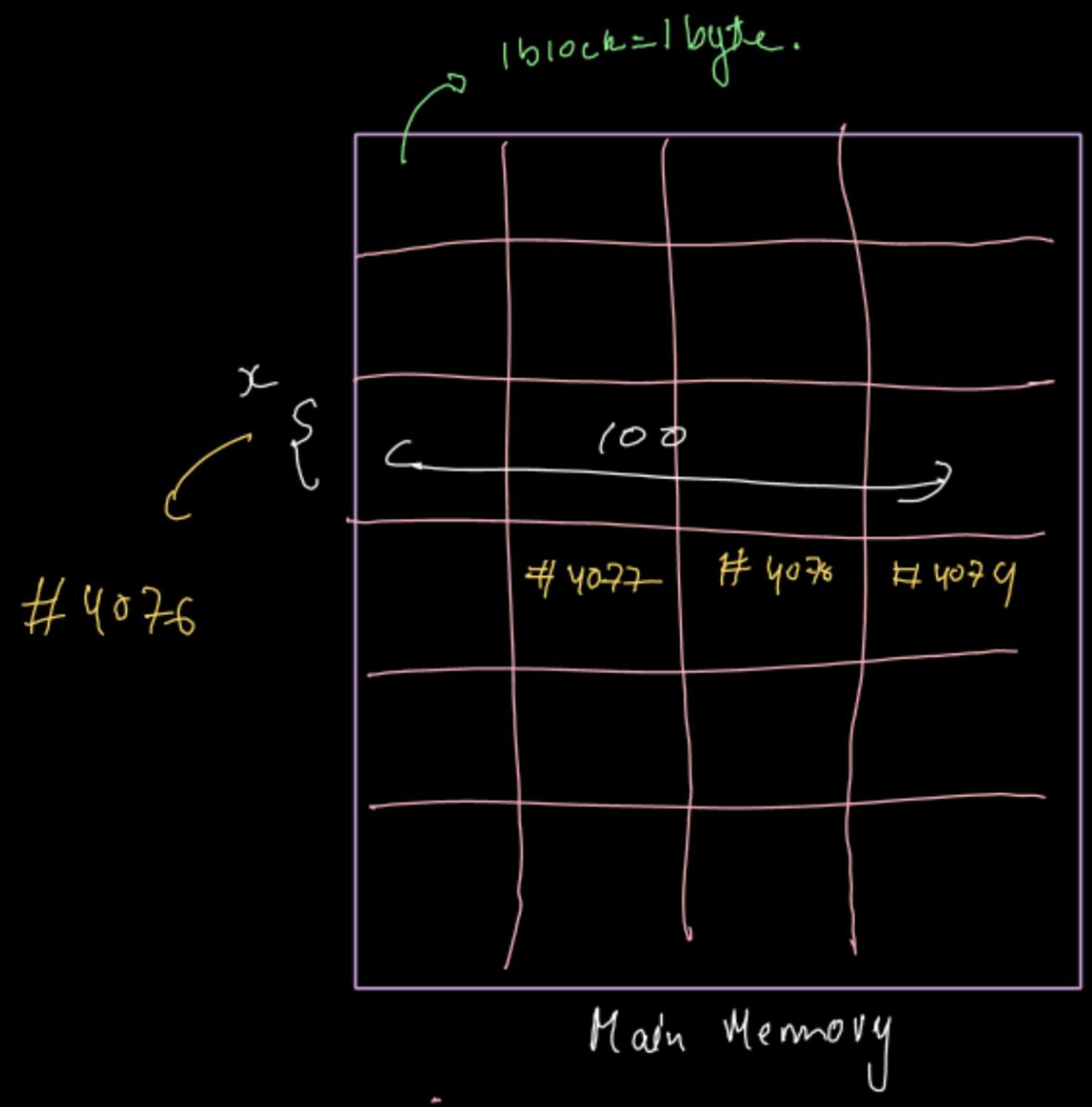
int x = 100;

datatype

constant

int x = 100;

```java
import java.util.*;
public class Main {
    public static void main(String[] args) {
        int x = 100;
        System.out.println(x);
    }
}
```



```java
import java.util.*;
public class Main {
    public static void main(String[] args) {
        int x = 100;
        System.out.println(x);

        char ch = 'a';
        System.out.println(ch);
    }
}
```

```
Finished in 64 ms
100
a
```

Operators in Java

```java
import java.util.*;
public class Main {
    public static void main(String[] args) {
        int x = 100;
        System.out.println(x);

        char ch = 'a';
        System.out.println(ch);
    }
}
```

int x = 100;

LHS RHS

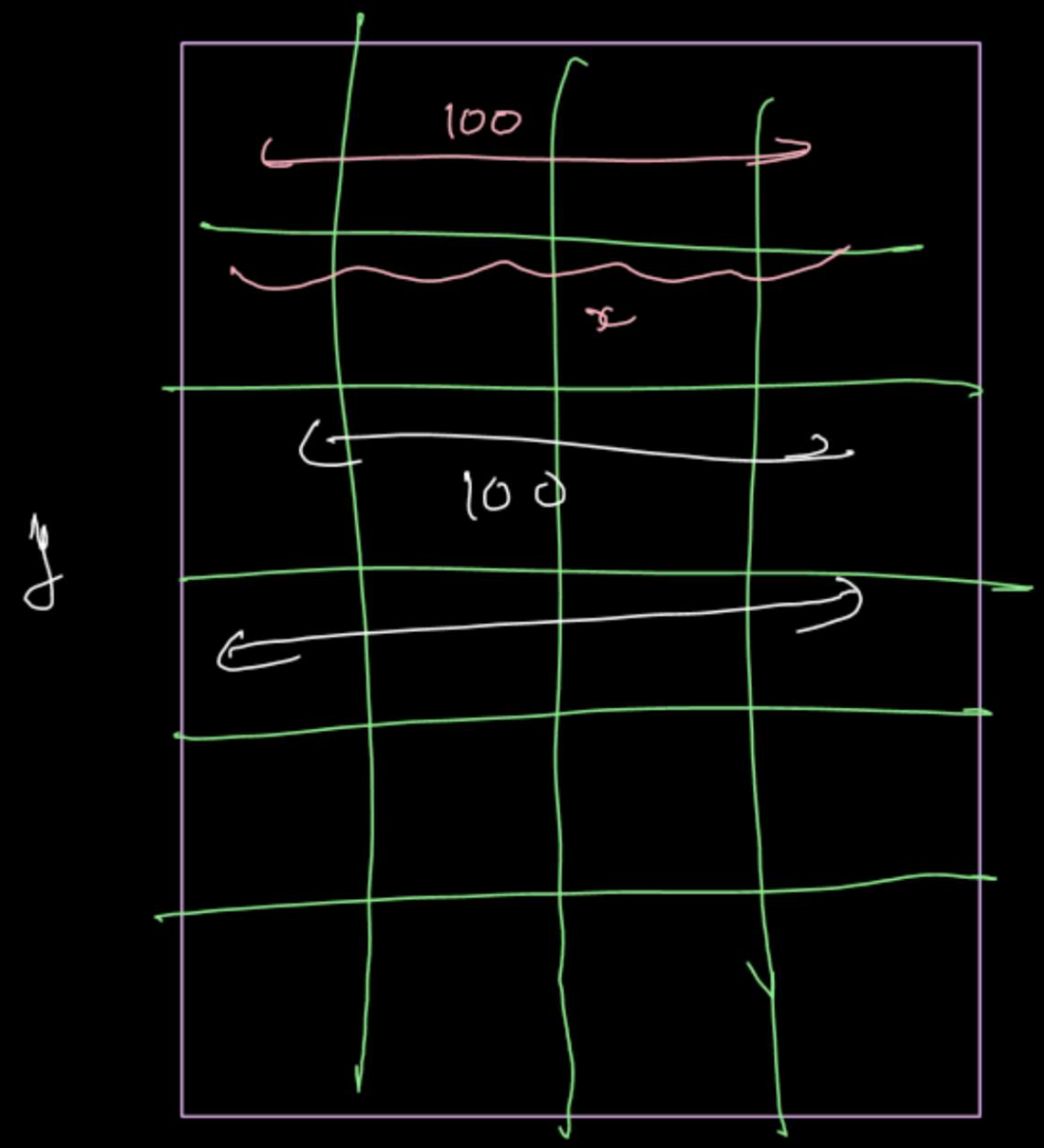
assignment operator

RHS gets assigned to LHS

```java
import java.util.*;
public class Main {
    public static void main(String[] args) {
        int x = 100;
        int y = x;
        System.out.println(x);
        System.out.println(y);
    }
}
```

int y = x;

int y = x;



```java
import java.util.*;
public class Main {
    public static void main(String[] args) {
        int x;
        x = 100;
        System.out.println(x);
    }
}
```

$x = \cancel{GN}\ 100$

```
Finished in 50 ms
100
```

* Initialization of values is a must in Java.

```java
import java.util.*;
public class Main {
    public static void main(String[] args) {
        int x;
        System.out.println(x);
    }
}
```

```
Finished in N/A
Line 5: error: variable x might not have been initialized [in Mai
n.java]
        System.out.println(x);
                           ^
```

Arithmetic Operators

+ plus

− minus

* multiply

/ divide

% modulus

```java
import java.util.*;
public class Main {
    public static void main(String[] args) {
        int x = 10;
        int y = 30;
        int res = x / y;
        System.out.println(res);
    }
}
```

10/30 = 1/3 = 0.33 . .

×

= 0

int x = 1.99

= 1

```java
import java.util.*;
public class Main {
    public static void main(String[] args) {
        int x = 10;
        int y = 30;
        float res = (float)x / y; //typecasting
        System.out.println(res);
    }
}
```

→ process of converting IDT to another.

integer operator → decimal truncates

x is converted to float 10 → 10.00

float / int = float

10/30 →

```
      0
30 ) 10 (
     -0
     10
```

→ Quotient is the ans of integer /.

10 → remainder is the ans of modulus.

```java
import java.util.*;
public class Main {
    public static void main(String[] args) {
        int x = 10;
        int y = 30;
        int res = x % y;
        System.out.println(res);
    }
}
```

```
Finished in 70 ms
10
```

```java
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int num = scn.nextInt();
        System.out.println(num);
    }
}
```

```
Finished in 108 ms
256
stdin
256
```

```java
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int num = scn.nextInt();
        System.out.println(num);

        float f = scn.nextFloat();
        System.out.println(f);
    }
}
```

```
Finished in 164 ms
256
10.987
```

```java
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int num = scn.nextInt();
        System.out.println(num);

    }
}
```

```
Finished in N/A

java.util.InputMismatchException
  at line 939, java.base/java.util.Scanner.throwFor
  at line 1594, java.base/java.util.Scanner.next
  at line 2258, java.base/java.util.Scanner.nextInt
  at line 2212, java.base/java.util.Scanner.nextInt
  at line 5, Main.main
```

stdin

```
10.987
```

```java
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        char ch = scn.nextChar();
        System.out.println(ch);
    }
}
```

```
Finished in N/A

Line 5: error: cannot find symbol [in Main.java]
        char ch = scn.nextChar();
                      ^
  symbol:   method nextChar()
  location: variable scn of type Scanner
```

Next Char is not a method.

Getting Started (Lecture : 2)

→ Conditional statements

- if - else
- if - else if - else ladder
- switch case

} comparison operators + logical operators

→ Loops

- for
- while
- do - while

} continue + break

→ functions in Java (with Memory concepts)

Questions

- Even or odd 1
- Even or odd 2
- factorial

Conditional Statements

```java
import java.util.*;
public class Main {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        boolean b = scn.nextBoolean();
        System.out.println(b);
    }
}
```

When we want the code (part of it) to execute in a certain condition.

#1 If - Else

```java
import java.util.*;
public class Main {

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

        System.out.println("Start");

        if(guessNo == 10) {
            System.out.println("You have won the game");
        } else {
            System.out.println("You have lost the game");
        }

        System.out.println("end");
    }
}
```

true

```
Finished in 153 ms
Start
You have won the game
end
```

stdin

10

$\boxed{\text{guessNo} == 10}$

LHS — comparison operator — RHS

if LHS = RHS

→ value of this statement is true

else it is false

```
if (condition) {
    ≡
}
else {
    ≡
}
```

true

if condition is true then this executes

} → if condition is false, then this executes

```java
import java.util.*;
public class Main {

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

        System.out.println("Start");

        if(guessNo >= 10 && guessNo <= 20) {
            System.out.println("You have won the game");
        } else {
            System.out.println("You have lost the game");
        }

        System.out.println("end");
    }
}
```

Greater than or equal to

Less than or equal to

logical AND

HW

> → Greater than

< → less than

AND

condition 1 **AND** condition 2

(&&)

condition 1 → T

condition 2 → T

then AND value will be true

else value is false

Logical OR

```java
import java.util.*;
public class Main {

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

        System.out.println("Start");

        if(guessNo == 10 || guessNo == 20) {
            System.out.println("You have won the game");
        } else {
            System.out.println("You have lost the game");
        }

        System.out.println("end");
    }
}
```

logical OR

cond 1 **OR** cond 2

if min one is true

answer is true

Logical Not

```java
import java.util.*;
public class Main {

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

        System.out.println("Start");

        if(!(guessNo == 10 || guessNo == 20)) {
            System.out.println("You have won the game");
        } else {
            System.out.println("You have lost the game");
        }

        System.out.println("end");
    }
}
```

Logical not

! (condition)

↳ it negates the condition

True → false

false → True

$$\left(!\left((\text{guessNo} == 10) \;||\; (\text{guessNo} == 20) \right) \right)$$

$$!\left((10) \;||\; (20) \right)$$

$$\times \qquad \times$$

Not Equal to

```java
import java.util.*;
public class Main {

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

        System.out.println("Start");

        if(guessNo != 10) {
            System.out.println("You have won the game");
        } else {
            System.out.println("You have lost the game");
        }

        System.out.println("end");
    }
}
```

Not equal to

LHS is NOT equal to RHS then statement value is true else false.

If – else if – else ladder

```java
import java.util.*;
public class Main {

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

        System.out.println("Start");

        if(guessNo == 10) {
            System.out.println("You have won the game");
        } else if(guessNo == 20) {
            System.out.println("You have won the game");
        } else {
            System.out.println("You have lost the game");
        }

        System.out.println("end");
    }
}
```

```
if (condn 1) {
    ==
} else if (condn 2) {
    ==
} else if (condn 3) {
    ==
}
else {
}
```

true

only this will execute

Multible Else if

```java
import java.util.*;
public class Main {

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

        System.out.println("Start");

        if(guessNo == 10) {
            System.out.println("You have won the game");
        } else if(guessNo == 20) {
            System.out.println("You have won the game");
        } else if(guessNo == 30) {
            System.out.println("You have won the game");
        }
        else {
            System.out.println("You have lost the game");
        }

        System.out.println("end");
    }
```

```
if (condition) {

}
```

```
if ( )
{
}
```

```
if (condn1) {

}
else if (condn2) {

}
if (condition 3) {

}
```

If there are multiple if statements one after another then they are checked.

```java
import java.util.*;
public class Main {

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

        System.out.println("Start");

        if(x == 10) {
            System.out.println("You have won the game");
        } else if(x == 20) {
            System.out.println("You have lost the game");
        }
        if(x == 100) {
            System.out.println("You have won the game");
        }

        System.out.println("end");
    }
}
```

```
Finished in 163 ms
Start
You have lost the game
end
```

stdin

```
20
100
```

for Loop

Print your name 10 times

```java
import java.util.*;
public class Main {

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

        for(int i=1;i<=10;i++) {
            System.out.println("DSA Classes");
        }
    }
}
```

```
Finished in 110 ms
DSA Classes
DSA Classes
DSA Classes
DSA Classes
DSA Classes
DSA Classes
DSA Classes
DSA Classes
DSA Classes
DSA Classes
```

Generally used when exact no of iterations is known.

initial condition

condition

```
for (int i = 1; i <= 10; i++) {
    Syso("DSA");
}
```

① ② ③

$i++ \rightarrow i = i + 1$

(1) + 1

(2)

$\boxed{\cancel{1}\ \cancel{2}}\ \cancel{3}\ \cancel{4} \ldots \cancel{9}\ \cancel{10}$

i

11

DSA
DSA
DSA
DSA
⋮

DSA
DSA

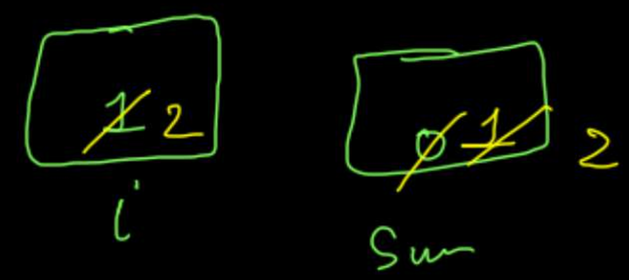
Counting 1 to N

```java
import java.util.*;
public class Main {

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

        int num = scn.nextInt();

        for(int i=1;i<=num;i++) {
            System.out.println(i);
        }
    }
}
```

Sum of 1 to N

num = 10

```java
import java.util.*;
public class Main {

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

        int num = scn.nextInt();
        int sum = 0;
        for(int i=1;i<=num;i++) {
            sum = sum + i;
        }

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



Sum = Sum + i

(1) + (2)

= 2

```java
import java.util.*;
public class Main {

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

        int num = scn.nextInt();
        int sum = 0;
        for(int i=1;i<=num;i++) {
            sum += i; //sum = sum + i
        }

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

While Loop

(when we don't know exact no of iterations. But, we know the condition)

```java
import java.util.*;
public class Main {

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

        int i = 1;
    ① while(i <= 10) {
            System.out.println("DSA Classes");
            i++;
        }
    }
}
```

initialization

condition

②

③

1 2 3 ... 11

i

DSA

DSA

⋮

DSA

Count N to 1

```java
import java.util.*;
public class Main {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        while(n >= 1) {
            System.out.println(n);
            n--;
        }
    }
}
```

$\boxed{\cancel{10}\ \cancel{9}}\ \cancel{8}\ \cancel{5}\ 0$

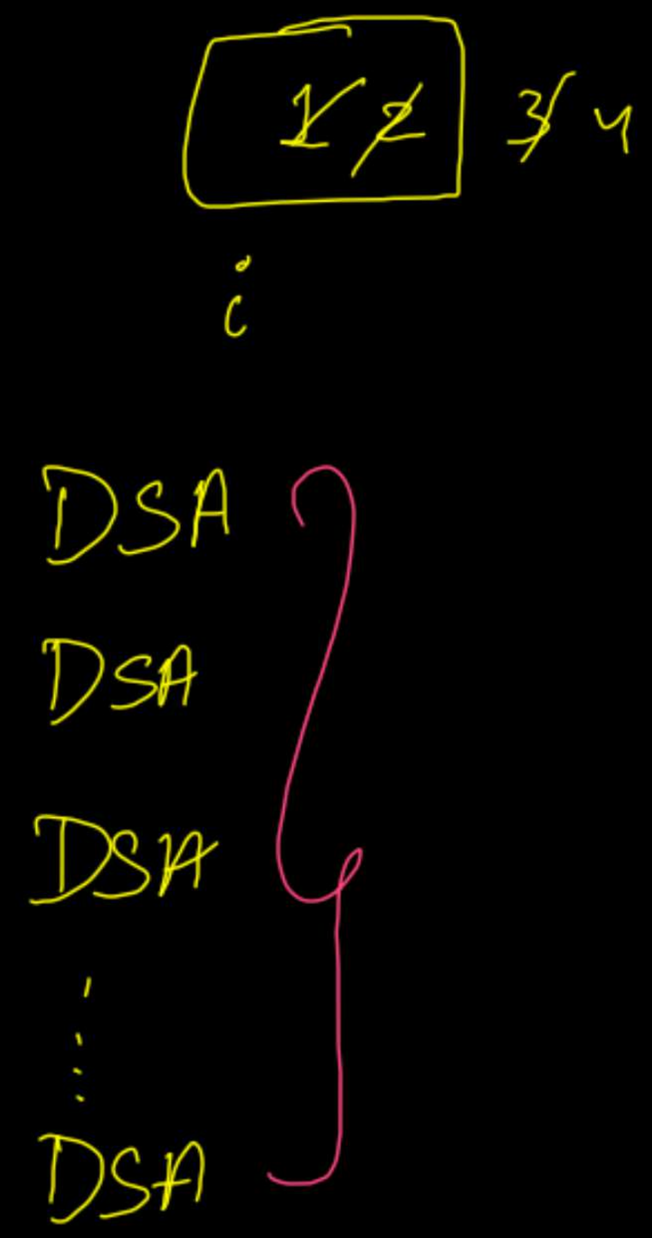
n

10
9
⋮
1

$n-- \rightarrow n = n - 1$

Do while Loop

```java
import java.util.*;
public class Main {

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

        int i = 1;  → initialization
        do{
            System.out.println("DSA");  → ①
            i++;  → ②
        } while(i<=10);  → ③
    }
}
```



```java
int i = 11;

while(i <= 10) {
    System.out.println("DSA");
    i++;
}
```

① 11 ≤ 10 → false

Loop not executed

```java
int i = 11;
do{
    System.out.println("DSA");
    i++;
} while(i<=10);
```

① DSA

i = ~~11~~ 12

12 ≤ 10 ✗

Continue in Loops

```java
import java.util.*;
public class Main {

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

        for(int i=1;i<=10;i++) {
            if(i % 3 == 0) {
                continue;
            }
            System.out.println(i);
        }
    }
}
```

① ③ ②

```
Finished in 119 ms
1
2
4
5
7
8
10
```

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

1
2
4

i % 3 == 0

skips the current iteration

Break in loops

```java
import java.util.*;
public class Main {

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

        for(int i=1;i<=10;i++) {
            if(i % 3 == 0) {
                break;
            }
            System.out.println(i);
        }
    }
}
```

break from the loop

i.e exit the loop.

Switch Case

```java
import java.util.*;
public class Main {

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

        switch(num) {
            case 10:
                System.out.println("Option 10");
                break;
            case 20:
                System.out.println("Option 20");
                break;
            case 30:
                System.out.println("Option 30");
                break;
            default:
                System.out.println("Please select out of 10, 20 or 30");
        }

```

or character or String

Getting Started - Lecture 3

1. Unary Operators
2. functions and their Memory Mappings

DSA Sheet Questions

1. Even or Odd - 1
2. Even or odd - 2
3. factorial
4. Sum or Product
5. Sum of Even Nos Till N

Unary Operators

Operand 1 operator Operand 2

+, -, *, /, %, && , || etc

operator Operand

++ → pre, post

-- → pre, post

Operand operator

+ (Unary Operator)

```java
import java.util.*;
public class Main {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int x = +3 + 5;
        System.out.println(x);

    }
}
```

Unary Binary

- (Unary Operator)

```java
import java.util.*;
public class Main {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int x = -3 + 5;
        System.out.println(x);

    }
}
```

Unary Binary

++ (Increment Operator)

Post

x++; ⟶ x = x + 1

OV + 1

Pre

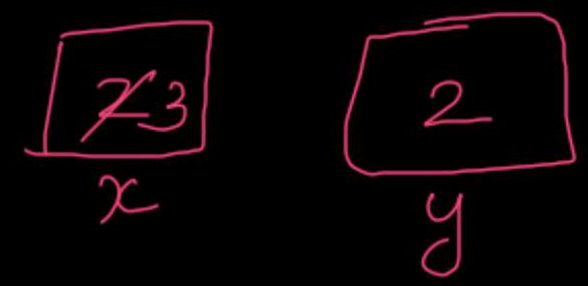
++x; ⟶ x = x + 1

y = x++ ⟶ True ⟵ y = ++x

2 statements are different.

Post Increment

```java
import java.util.*;
public class Main {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int x = 2;
        int y = x++;
        System.out.println(x);
        System.out.println(y);
    }
}
```



y = x++;

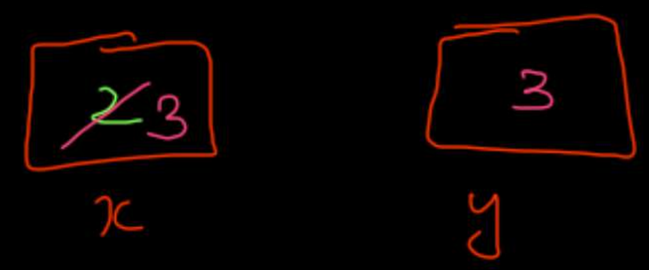
assignment increment

① y = x

② x = x + 1

Pre Increment

```java
import java.util.*;
public class Main {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int x = 2;
        int y = ++x;
        System.out.println(x);
        System.out.println(y);
    }
}
```



$y = ++x;$

① $x = x + 1$

② $y = x$

Post Decrement

```java
import java.util.*;
public class Main {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int x = 2;
        int y = x--;
        System.out.println(x);
        System.out.println(y);
    }
}
```

$\boxed{\cancel{2}\,1}$ $\boxed{2}$

x y

$y = x--;$

① $y = x$

② $x = x - 1$

$x--;$

$\hookrightarrow x = x - 1;$

Pre decrement

```java
import java.util.*;
public class Main {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int x = 2;
        int y = --x;
        System.out.println(x);
        System.out.println(y);
    }
}
```

$\boxed{\cancel{2}\,1}$ x

$\boxed{1}$ y

y = --x;

① x = x - 1

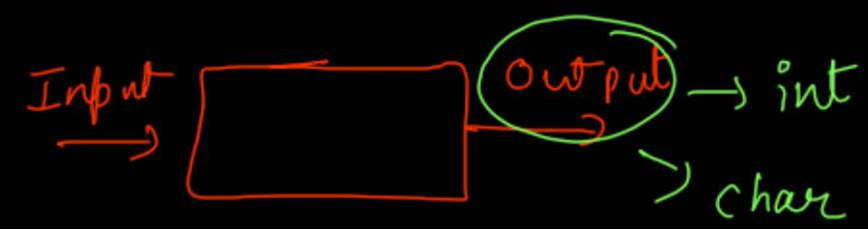
② y = x

functions

```java
import java.util.*;
public class Main {

    public static void counting() {
        //code
    }

    public static void main(String[] args) {
        //some code
    }
}
```

name

Return type



datatype

void means function does not return anything.

```java
import java.util.*;
public class Main {

    public static void counting(int n) {
        //code
    }

    public static void main(String[] args) {
        //some code
    }
}
```

parameter

```java
import java.util.*;
public class Main {

    public static void counting(int n) {
        //code
        for(int i=1;i<=n;i++) {
            System.out.println(i);
        }
    }

    public static void main(String[] args) {
        //function call
        counting(10);

    }
}
```

function body

function call

```java
import java.util.*;
public class Main {

    public static void counting(int n) {
        //code
        for(int i=1;i<=n;i++) {
            System.out.println(i);
        }
    }

    public static void main(String[] args) {
        //function call
        counting(10);
        System.out.println();
        counting(20);
    }
}
```

Execution of Java

program starts from

the main method (function)

```java
import java.util.*;
public class Main {
    public static void counting(int n) {
        //code
        for(int i=1;i<=n;i++) {
            System.out.println(i);
        }
    }

    public static void main(String[] args) {
        //function call
        int x = 10;
        counting(x);
        System.out.println();
    }
}
```

function definition

parameter

argument

① ② ③

Pause

counting

10

n

main

10

x

Stack

Heap

(free pool of Memory)

1
2
3
4
5
6
7
8
9
10

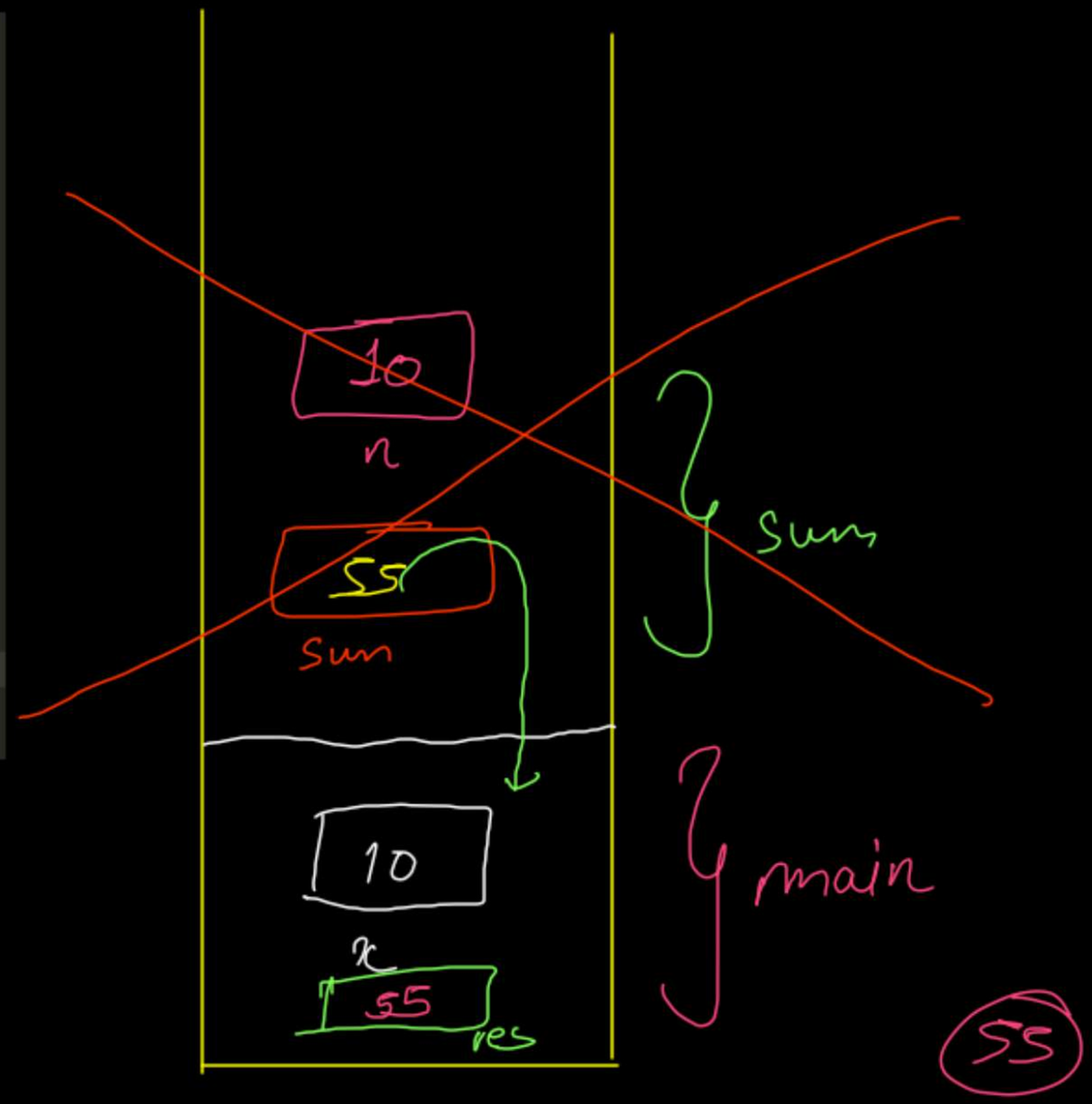
function call Stack

① All primitive datatypes (8) are allocated memory inside the Stack.

② As soon as all the lines of code of a function are executed, it gets wiped out of the stack.

③ In Java, functions are called Methods because in OOPS any function inside a class is called a Method & in Java everything is inside a class.

```java
import java.util.*;
public class Main {

    public static int sum(int n) {
        //function body
        int sum = 0;
        for(int i=1;i<=n;i++) {
            sum += i;
        }
        return sum;
    }

    public static void main(String[] args) {
        //function call
        int x = 10;
        int res = sum(10);
        System.out.println(res);
    }
}
```



```java
import java.util.*;
public class Main {

    public static int sum(int n) {
        //function body
        int sum = 0;
        for(int i=1;i<=n;i++) {
            sum += i;
        }
        return sum;
    }

    public static void main(String[] args) {
        //function call
        int x = 10;
        System.out.println(sum(10));
    }
}
```

55

Even Odd

Easy Accuracy: **48.64%** Submissions: **26932** Points: **2**

We've learnt about operators and other basics of CPP. Now, it's time to take another leap and learn to use control structures that helps us choose flow of any code.

Given two integers **a** and **b**. Your task is to print the **even number first and odd number next** in individual lines.
Note: Must print end of the line at the end.

```java
//User function Template for Java

class Solution{
    public void evenOdd(int a, int b){
        // Code here
        if(a % 2 == 0) {
            System.out.println(a);
            System.out.println(b);
        } else {
            System.out.println(b);
            System.out.println(a);
        }
    }
}
```

Even No

$$\boxed{N \% 2 = 0}$$

odd No

$$\boxed{N \% 2 \; ! = 0 \quad (= 1)}$$

Odd or Even

School Accuracy: **60.31%** Submissions: **13337** Points: **0**

Given a positive integer N, determine whether it is odd or even.

Example 1:

Input:
N = 1
Output:
odd
Explanation:
The output is self-explanatory.

```java
//User function Template for Java
class Solution{
    static String oddEven(int N){
        // code here
        if(N % 2 == 0) {
            return "even";
        } else {
            return "odd";
        }
    }
}
```

System.out.print/ln("↓");

This is a String

Factorial

Basic Accuracy: **53.69%** Submissions: **34040** Points: **1**

Given a positive integer, **N**. Find the factorial of **N**.

Example 1:

Input:
N = 5
Output:
120
Explanation:
5*4*3*2*1 = 120

```
class Solution{
    static long factorial(int n){
        // code here
        if(n == 0) return 1L;
        long product = 1L;
        for(int i=n;i>=1;i--) {
            product = product * i;
        }

        return product;
    }
}
```

factorial

$N! = N*(N-1)*(N-2)\ldots 1$

$= 1*2\times 3 \ldots\ldots *(N-1)*(N-2)$

$\boxed{\cancel{5}\ 4}$ $\boxed{\cancel{1}\ \cancel{3}}$ $\cancel{20}$ $\cancel{60}$ 120

i product

$p = p * i$

(3) * (4)

Problem | Submissions | Solution | New Discuss

Sum or Product

1006 Difficulty: EASY

Contributed By
Ankush Gupta | Level 1

Avg. time to solve
15 min

Success Rate
80%

Problem Statement

Suggest Edit

You are given a number 'N' and a query 'Q.' If 'Q' is 1, then you have to return the sum of all integers from 1 to 'N,' else if 'Q' is equal to 2 then you have to return the product of all integers from 1 to 'N.' Since the product can be very large, return it modulo 10 ^ 9 + 7.

For example

```
Given 'N' = 4, 'Q' = 1.
Then the answer is 10 because the sum of all integers between 1 and 4 are 1, 2, 3, and
4. Hence 1 + 2 + 3 + 4 is equal to 10.
```

Sum of 1st N natural No

$N(N+1)/2$

$10^9+7 = (1000000007)$

0 to 10^9+6

$7 \overline{)1}$

-0

1

$7 \overline{)8}$

-7

1

$1 \% 7 = 1$

$2 \% 7 = 2$

$3 \% 7 = 3$

$4 \% 7 = 4$

$5 \% 7 = 5$

$6 \% 7 = 6$

$7 \% 7 = 0$

$8 \% 7 = 1$

$9 \% 7 = 2$

$\vdots$

$10^9+7 \rightarrow$ 1, 2, 3, 4, ...

10^9+8

10^9+9

$\vdots$

10^9+6

10^9+6

INT * INT

(∞) $(\infty-1)$ $\rightarrow$ out of Range

$N*(N-1) \ldots 1$

```java
public class Solution {

    public static long sum(int n) {
        return (long)((n*(n+1))/2);
    }

    public static long factorial(int n) {
        if(n == 0) return 1L;
        long product = 1L;
        for(int i=n;i>=1;i--) {
            product = (product * i) % 1000000007;
        }

        return product;
    }
```

```java
    public static long sumOrProduct(int n, int q) {

        if(q == 1) {
            //sum
            return sum(n);
        } else {
            //factorial
            return factorial(n);
        }
    }
```

Getting Started - Lecture 4

Basic Syntax

- → Sum of Even Nos Till N
- → Swap 2 Nos **
  - → 3rd variable
  - → w/o 3rd variable
- → Switch Case

→ # Java is always pass by value

Basic Maths

- → Check Prime
  - → N
  - → N/2
  - → $\sqrt{N}$
- → Prime Nos in Range
- → GCD & LCM
  - → Normal Loop
  - → Euclidean
- → N^{th} fibonacci No
- → Bulb Switcher
- → Power of 2

71

Sum Of Even Numbers Till N

Difficulty: EASY

Contributed By
Ayush Thakur | Level 1

Avg. time to solve
10 min

Success Rate
90%

Problem Statement Suggest Edit

You have been given a number 'N'. Your task is to find the sum of all even numbers from 1 to 'N' (both inclusive).

Example :

```
Given 'N' : 6
Sum of all even numbers till 'N' will be : 2 + 4 + 6 = 12
```

10

1 2 3 4 5 6 7 8 9 10

2+4+6+8+10 → ans

if (condn) {

}

```
public class Solution {
    public static long evenSumTillN(int n) {
        // Write your code here.
        long sum = 0;
        for(int i=1;i<=n;i++) {
            if(i % 2 == 0) sum += i;
        }

        return sum;
    }
}
```

Swap two numbers

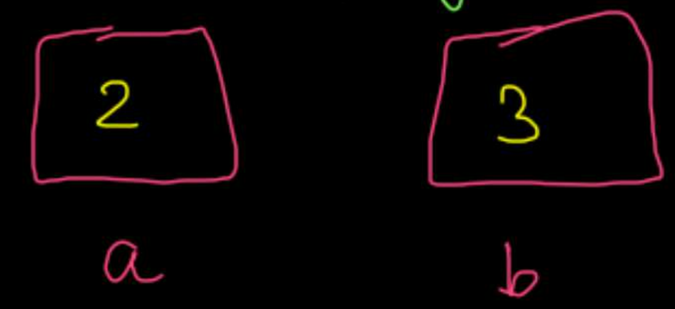
School Accuracy: **69.58%** Submissions: **14283** Points: **0**

Swap given two numbers and print them. (Try to do it without a temporary variable.) and return it.

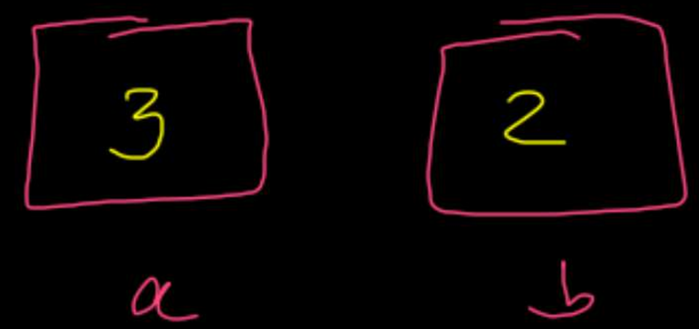
Example 1:

Input: a = 13, b = 9
Output: 9 13
Explanation: after swapping it becomes 9 and 13.

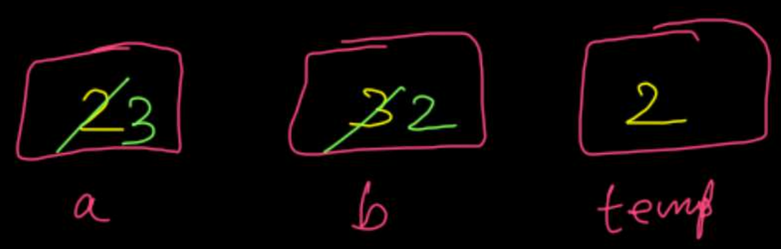
Before Swapping



After Swapping



Using 3rd variable



① int temp = a;

② a = b;

③ b = temp;

Without 3rd Variable

a	b
~~2~~ 5	~~3~~ 2

① $a = a + b$

(2) (3) = (5)

② $b = a - b$;

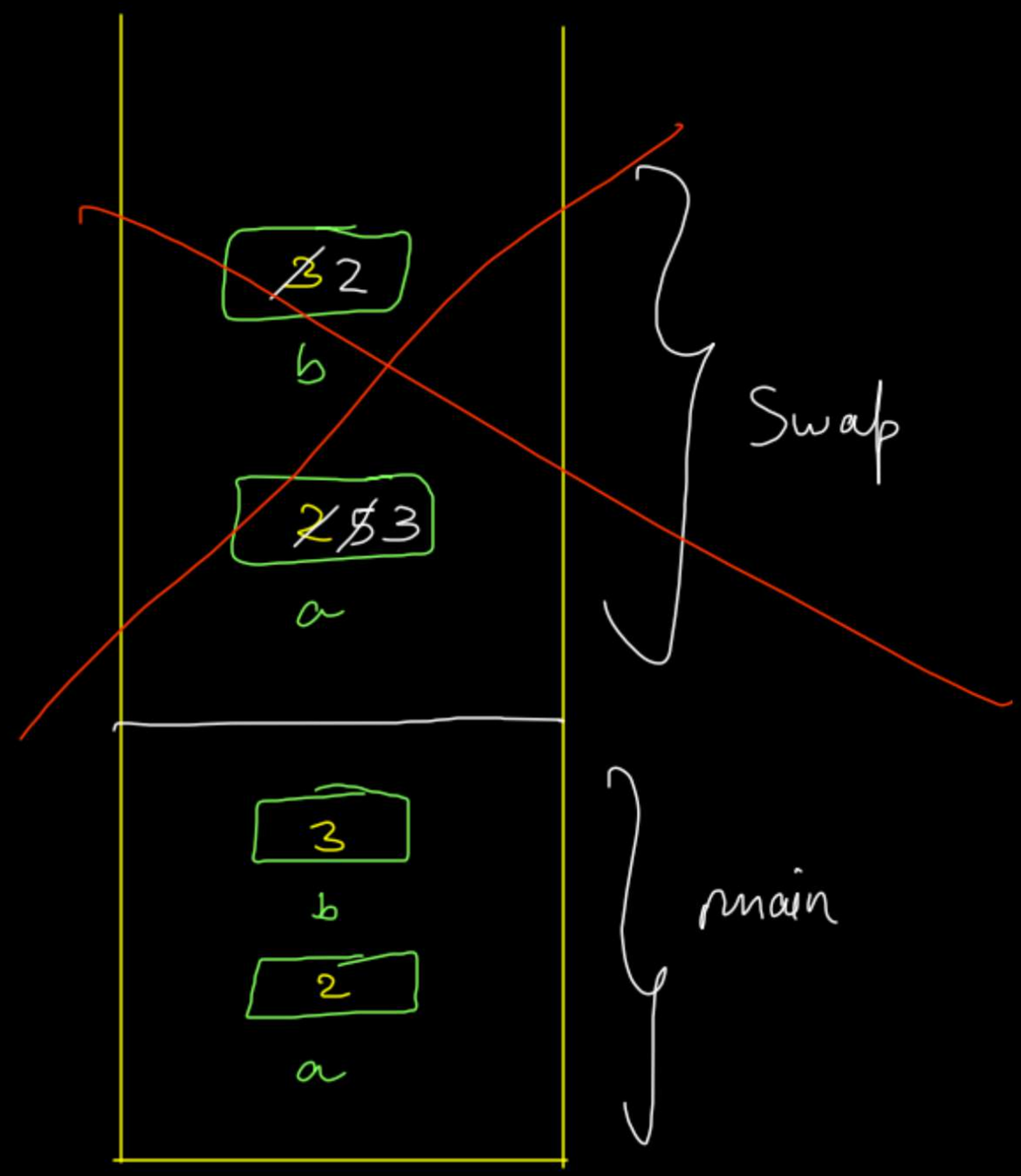
(5) (3) = (2)

③ $a = a - b$;

(5) - (2) = (3)

```java
import java.util.*;
public class Main {

    public static void swap(int a, int b) {
        //swapping
        a = a + b;
        b = a - b;
        a = a - b;
    }

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

        System.out.println("Before swapping a = " + a + " b = " + b);
        swap(a,b);
        System.out.println("After swapping a = " + a + " b = " + b);
    }
}
```

Pass by value



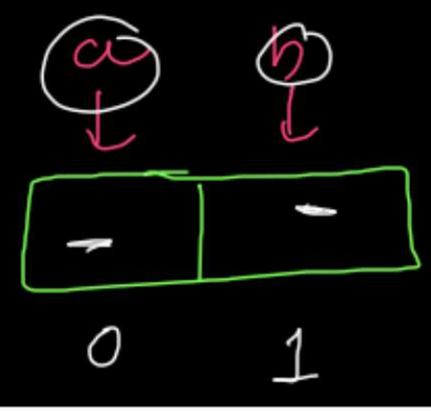
* Java is always pass by value

① Swap with 3rd variable

② W/o 3rd variable

③

④ Put the nos in a data structure (Array, List, LL etc.)



→ list (changes will reflect across function)

```java
class Solution{
    static List<Integer> get(int a,int b)
    {
        // code here
        a = a + b;
        b = a - b;
        a = a - b;

        List<Integer> list = new ArrayList<>();

        list.add(a);
        list.add(b);

        return list;
    }
}
```

Java Switch Case statement

School Accuracy: **66.06%** Submissions: **2535** Points: **0**

Given an integer choice denoting the choice of the user and a list containing the single value R or two values L and B depending on the choice.
If the user's choice is 1, calculate the area of the circle having the given radius(**R**).
Else if choice is 2, calculate the area of the rectangle with given length(**L**) and breadth(**B**).

```java
class Solution{
    static double switchCase(int choice, List<Double> arr){
        // code here

        double r = arr.get(0);
        double l = r;
        double b = (arr.size() == 2) ? arr.get(1) : 0;

        switch(choice) {
            case 1:
                return Math.PI *r *r;
            case 2:
                return l * b;
            default:
            return 0;
        }
    }
}
```

Ternary Operator

```java
import java.util.*;
public class Main {

    public static void swap(int a, int b) {
        //swapping
        a = a + b;
        b = a - b;
        a = a - b;
    }

    public static void main(String[] args) {
        boolean b = true;
        int a = (b == true) ? 1 : 0;
        System.out.println(a);
    }
}
```

(condition) ? if true : if false

Prime Number

Basic Accuracy: **49.19%** Submissions: **56391** Points: **1**

For a given number **N** check if it is prime or not. A prime number is a number which is only **divisible by 1 and itself**.

Example 1:

Input:
N = 5
Output:
1
Explanation:
5 has 2 factors 1 and 5 only.

2 → 2, 1

3 → 3, 1

4 → 2, 4, 1 ✗

5 → 5, 1

6 → 2, 3, 6, 1 ✗

7 → 2, 7

⋮

1 2 3 4 . . . . . N-1 N

loop & check

24 1, 24

29 1, 29

```
class Solution{
    static int isPrime(int N){
        // code here
        if(N == 1) return 0;

        for(int i=2;i<N;i++) {
            if(N % i == 0) return 0;
        }

        return 1;
    }
}
```

→ This code gives TLE.

O(N) solution

27 and 24

27 → 9 less than N/2

24 → 12 equal to N/2

$1 \times 27 = 27$

$3 \times 9 = 27$

1, 3, 9, 27

$1 \times 24 = 24$

$2 \times 12 = 24$

$3 \times 8 = 24$

$4 \times 6 = 24$

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

24 / 2 = 12

smallest * largest → N/2 is the largest factor in case of even Nos

(2) (12)

25

smallest * largest

```java
class Solution{
    static int isPrime(int N){
        // code here
        if(N == 1) return 0;

        for(int i=2;i<N/2;i++) {
            if(N % i == 0) return 0;
        }

        return 1;
    }
}
```

iterations have reduced to half.

O(N)

Square Root Solution

36 → 6

$1 \times 36 = 36$

$2 \times 18 = 36$

$3 \times 12 = 36$

$4 \times 9 = 36$

$6 \times 6 = 36$

1, 2, 3, 4, 6, 9, 12, 18, 36

24 → 4

$1 \times 24 = 24$

$2 \times 12 = 24$

$3 \times 8 = 24$

$4 \times 6 = 24$

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

25

$1 \times 25 = 25$

$5 \times 5 = 25$

1, 5, 25

29 → 5

$1 \times 29 = 29$

1, 29

5 → Since there are No factors equal to or before 5 (i.e. $\sqrt{N}$) hence, 29 is a prime No.

Code conversion

$i \le \sqrt{N}$

Squaring both sides

$i^2 \le N$

$\Rightarrow \boxed{i * i \le N}$

625

2 to 624 1st Method

2 to 317 2nd Method

2 to 25 3rd Method

```java
class Solution{
    static int isPrime(int N){
        // code here
        if(N == 1) return 0;

        for(int i=2;i * i <=N;i++) {
            if(N % i == 0) return 0;
        }

        return 1;
    }
}
```

$\# \; O(\sqrt{N})$

Find Prime numbers in a range

Medium Accuracy: **63.56%** Submissions: **17866** Points: **4**

Given two integers M and N, generate all primes between M and N including M and N.

```java
boolean isPrime(int n) {

    if(n == 1) return false;

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

    return true;
}
```

```java
ArrayList<Integer> primeRange(int lo, int hi) {
    // code here

    ArrayList<Integer> res = new ArrayList<>();

    for(int j=lo;j<=hi;j++) {
        if(isPrime(j) == true) res.add(j);
    }

    return res;
}
```

319. Bulb Switcher

Medium 1033 1751 Add to List Share

There are `n` bulbs that are initially off. You first turn on all the bulbs, then you turn off every second bulb.

On the third round, you toggle every third bulb (turning on if it's off or turning off if it's on). For the `i^th` round, you toggle every `i` bulb. For the `n^th` round, you only toggle the last bulb.

Return *the number of bulbs that are on after* `n` *rounds.*

1 ✓
2 ✗
3 ✗
4 ✗ ✓
5 ✗
6 ✗ ✗
7 ✗
8 ✗ ✗
9 ✗ ✓
10 ✗ ✗
11 ✗
12 ✗ ✗ ✗
13 ✗
14 ✗ ✗

15 ✗ ✗
16 ✗ ✗ ✓
17 ✗
18 ✗ ✗ ✗
19 ✗
20 ✗ ✗ ✗
21 ✗ ✗
22 ✗ ✗
23 ✗
24 ✗ ✗ ✗ ✗
25 ✗ ✓

✓ → ON
✗ → off

24 → 4

$1 \times 24 = 24$

$2 \times 12 = 24$

$3 \times 8 = 24$

$4 \times 6 = 24$

$6 \times 4 = 24$

$8 \times 3 = 24$

$12 \times 2 = 24$

$24 \times 1 = 24$

36 → 6

$1 \times 36 = 36$

$2 \times 18 = 36$

$3 \times 12 = 36$

$4 \times 9 = 36$

$6 \times 6 = 36$ → This does not mirror.

$9 \times 4 = 36$

$12 \times 3 = 36$

$18 \times 2 = 36$

$36 \times 1 = 36$

Bulb Switcher

```java
class Solution {
    public int bulbSwitch(int n) {

        int count = 0;
        for(int i=1;i<=n;i++) {
            int sqrt = (int)Math.sqrt(i);
            if(sqrt * sqrt == i) count++;
        }

        return count;
    }
}
```

Power of 2

```java
class Solution {
    public boolean isPowerOfTwo(int n) {
        if(n < 0) return false;

        for(int i=0;i<=30;i++) {
            int power = (int)Math.pow(2,i);
            if(power == n) return true;
        }

        return false;
    }
}
```

negative nos cannot be represented as

$n = 2^x$

Nth Fibonacci Number

Easy Accuracy: **41.85%** Submissions: **76409** Points: **2**

Given a positive integer n, find the nth fibonacci number. Since the answer can be very large, return the answer modulo 1000000007.

Example 1:

Input: n = 2
Output: 1
Explanation: 1 is the 2nd number of fibonacci series.

Example 2:

0th	1st	2nd	3rd	4th	5th ...
0	1	1	2	3	5 ...

** Every term is the sum of previous 2

first = 0 (if N==0)

second = 1 (if N==1)

third = 0

```
for(int i=2; i<=N; i++)
{   third = first + second;
    first = second;
    second = third;
}
```

```
//User function Template for Java
class Solution {
    static long nthFibonacci(long n){
        // code here

        long a = 0L;
        long b = 1L;
        long c = 0L;

        for(int i=2;i<=n;i++) {
            c = (a + b) % 1000000007;
            a = b;
            b = c;
        }

        return c;
    }
}
```

5th

a = ~~0~~ ~~1~~ ~~1~~ ~~2~~ 3

b = ~~1~~ ~~1~~ ~~2~~ ~~3~~ 5

c = ~~0~~ ~~1~~ ~~2~~ ~~3~~ 5

i = ~~2~~ ~~3~~ ~~4~~ ~~5~~

5

c = a+b

2+3 = 5

0 1 1

This is DP Space

Optimized Approach.

LCM And GCD

Basic Accuracy: **55.13%** Submissions: **14347** Points: **1**

Given two numbers **A** and **B.** The task is to find out their LCM and GCD.

24, 36

~~23~~

~~22~~

~~21~~

⋮

12 → HCF

$$A * B = LCM * GCD$$

$$LCM = (A * B) / GCD$$

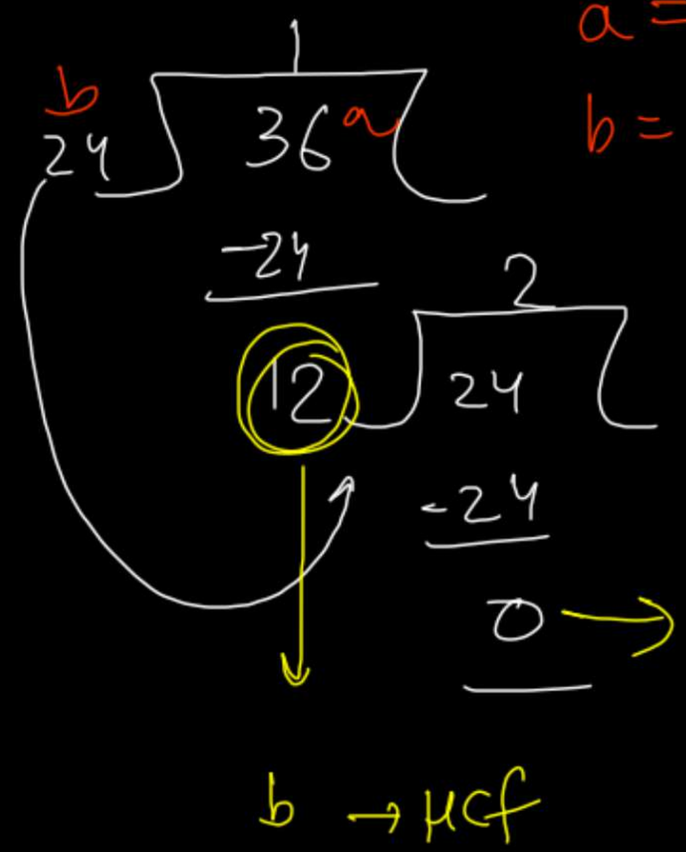
```java
static long gcd(long a,long b) {

    long originalA = a;

    while(a >= 1) {
        if(originalA % a == 0 && b % a == 0) return a;
        a--;
    }

    return 1L;
}
```

→ TLE

```java
static Long[] lcmAndGcd(Long A , Long B) {
    // code here

    long gcd = gcd(A,B);
    long lcm = (A*B)/gcd;

    Long[] res = new Long[2];
    res[0] = lcm;
    res[1] = gcd;
    return res;
}
```

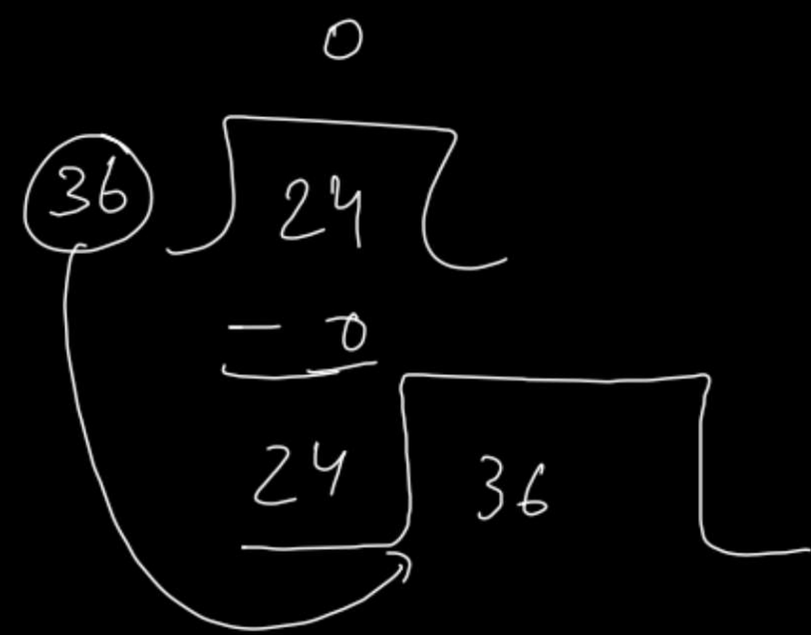
Euclidean Method

24 and 36



a = b

b = rem




```java
static long divisionMethod(long a, long b) {
    while(a % b != 0) {
        long rem = a%b;
        a = b;
        b = rem;
    }

    return b;
}
```

Getting Started - Lecture 4

Digit Traversal

1. Digits of a Number
2. Count Digits of a Number (Condition Based)
3. Subtract Product & Sum of Digits
4. Rotate a Number
5. Reverse a Number
6. Palindrome Number
7. Armstrong Number
8. Inverse of a Number.

Digits of a Number

Reverse A Number

Easy

Prev Next

1. You've to display the digits of a number in reverse.
2. Take as input "n", the number for which digits have to be display in reverse.
3. Print the digits of the number line-wise, but in reverse order.

Input Format

"n" where n is any integer.

Input Format

"n" where n is any integer.

Output Format

d1
d2
d3
... digits of the number in reverse

65784383

3
8
3
4
8
7
5
6

$65784383 \longrightarrow n$

```
int digit = n % 10;

n = n/10;
```

$$
\begin{array}{r}
6578438 \\
10 \overline{)\,65784383} \\
-60 \\
\hline
57 \\
-50 \\
\hline
78 \\
-70 \\
\hline
84 \\
-80 \\
\hline
43 \\
-40 \\
\hline
38 \\
-30 \\
\hline
\end{array}
$$

$83 \,/\, -80 = 3$

	digit (N % 10)	N/10
65784383	3	6578438
6578438	8	657843
657843	3	65784
65784	4	6578
6578	8	657
657	7	65
65	5	6
6	6	0
0 → end point		

6/10

= 0.6

$N = 0 \rightarrow 0$ # corner case

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

    int n = scn.nextInt();

    //digits of a number

    if(n == 0) {
        System.out.println(0);
        return;
    }


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


    }
}
```

$\} \rightarrow O(\log_{10} N)$

Count Digits

School Accuracy: **42.91%** Submissions: **9265** Points: **0**

Given a number **N**. Count the number of digits in N which evenly divides N.

Example 1:

Input:
N = 12
Output:
2
Explanation:
1, 2 both divide 12 evenly

N = 124

N	digits	N / 10	original N (124)
124	4	12	count = ~~0~~ 1
12	2	1	count = ~~1~~ 2
1	1	0	count = ~~2~~ 3

980234

→ 98234 % 0

∞

Arithmetic Exception

↳ / by zero exception

```
class Solution{
    static int evenlyDivides(int N){
        // code here
        int originalN = N;
        int count = 0;

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

        return count;
    }
}
```

→ $O(\log_2 N)$

Rotate A Number

Easy

Prev Next

1. You are given two numbers n and k. You are required to rotate n, k times to the right. If k is positive, rotate to the right i.e. remove rightmost digit and make it leftmost. Do the reverse for negative value of k. Also k can have an absolute value larger than number of digits in n.
2. Take as input n and k.
3. Print the rotated number.
4. Note - Assume that the number of rotations will not cause leading 0's in the result. e.g. such an input will not be given

   n = 12340056

   k = 3

   r = 05612340

562984 → N → mod = 6

→ k (No of rotations)

k % mod ?

(0) k = 0 562984 ⟷ k = 6 562984 (0)

(1) k = 1 456298 ⟷ k = 7 456298 (1)

k = 2 845629

k = 3 984562

k = 4 298456

k = 5 629845

k = 8

k = 9

k = 10

k = 11

$$
\begin{array}{r}
166 \\
6\,\overline{)1000} \\
-6\downarrow \\
\hline
40 \\
-36 \\
\hline
40 \\
-36 \\
\hline
4
\end{array}
$$

✓

① $k = k \% nod$ (To convert large values of k to small values)

(0 to Nod-1)

Nod = 6

#② if (k < 0)

$k = k + nod$

k = 0 562984

k = 1 456298 ⟷ k = -5 456298

k = 2 845629 ⟷ k = -4 845629

k = 3 984562 ⟷ k = -3 984562

k = 4 298456 ⟷ k = -2 298456

k = 5 629845 ⟷ k = -1 629845

$N = 562984$

$K = 2$

845629

$$
\begin{array}{r}
5629 \\
100 \overline{)562984} \\
-500 \\
\hline
629 \\
-600 \\
\hline
298 \\
-200 \\
\hline
984 \\
-900 \\
\hline
84
\end{array}
$$

$Last = N \% 10^k \quad (84)$

$N = N / 10^k \quad (5629)$

$Last = Last * 10^{nod-k} \quad (840000)$

$N = Last + N \quad (845629)$

```java
public static void main(String[] args) {
    // write your code here

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

    int temp = n;
    int nod = 0;
    while(temp > 0) {
        nod++;
        temp /= 10;
    }

    k = k % nod;
    if(k < 0) k = k + nod;

    int last = n % (int)Math.pow(10,k);
    n = n / (int)Math.pow(10,k);
    last = last * (int)Math.pow(10,nod-k);
    n = n + last;

    System.out.println(n);

    }
}
```

$-1000 \cancel{k}$

$k = k \% nod$

$-(1000 \% 6)$

$k = -4$

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

Easy 1506 196 Add to List Share

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

Example 1:

```
Input: n = 234
Output: 15
Explanation:
Product of digits = 2 * 3 * 4 = 24
Sum of digits = 2 + 3 + 4 = 9
Result = 24 - 9 = 15
```

No = 234

N	digits	Sum (0)	Product (1)
234	4	4	4
23	3	7	12
2	2	9	24
0			

24 - 9 = 15

```java
class Solution {
    public int subtractProductAndSum(int n) {
        int sum = 0;
        int product = 1;

        while(n > 0) {
            int digit = n % 10;
            sum += digit;
            product *= digit;
            n /= 10;
        }

        return product - sum;
    }
}
```

Armstrong Numbers

School Accuracy: **53.34%** Submissions: **21079** Points: **0**

For a given **3** digit number, find whether it is armstrong number or not. An **Armstrong number** of three digits is an integer such that the sum of the cubes of its digits is equal to the **number** itself. Return "**Yes**" if it is a armstrong number else return "**No**".

NOTE: 371 is an **Armstrong number** since $3^3 + 7^3 + 1^3 = 371$

on = n

n = 371 ans

N	digit	ans (ans += d*d*d)
371	1	1
37	7	1+(7*7*7) = 344
3	3	344+(3*3*3) = 371
0		

```java
//User function Template for Java
class Solution {
    static String armstrongNumber(int n){
        // code here

        int originalN = n;
        int ans = 0;
        while(n > 0) {
            int d = n % 10;
            n/=10;
            ans += d*d*d;
        }

        if(ans == originalN) return "Yes";
        return "No";
    }
}
```

7. Reverse Integer

Medium 8234 10570 Add to List Share

Given a signed 32-bit integer `x`, return `x` *with its digits reversed.* If reversing `x` causes the value to go outside the signed 32-bit integer range `[-2^31, 2^31 - 1]`, then return `0`.

Assume the environment does not allow you to store 64-bit integers (signed or unsigned).

Example 1:

```
Input: x = 123
Output: 321
```

Example 2:

```
Input: x = -123
Output: -321
```

1234

rev = 0

0

rev = (rev * 10) + digit

No	digit	N/10	Rev
1234	4	123	4
123	3	12	43
12	2	1	432
1	1	0	4321

```java
class Solution {
    public int reverse(int x) {
        int rev = 0;

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

            rev = (rev * 10) + digit;
            x /= 10;
        }

        return rev;
    }
}
```

Integer

$\infty * 10 \rightarrow$ out of Range

$-\infty * 10 \rightarrow$ out of Range

$rev * 10 > Integer.MAX\ value$

$rev > \frac{Integer.MAX_VALUE}{10}$

$rev * 10 < Integer.MIN_VALUE$

$rev < \frac{Integer.MIN_VALUE}{10}$

```java
class Solution {
    public int reverse(int x) {
        int rev = 0;

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

            if((rev > Integer.MAX_VALUE/10) || (rev < Integer.MIN_VALUE/10))
                return 0;

            rev = (rev * 10) + digit;
            x /= 10;
        }

        return rev;
    }
}
```

9. Palindrome Number

Easy 7012 2274 Add to List Share

Given an integer `x`, return `true` if `x` is palindrome integer.

An integer is a **palindrome** when it reads the same backward as forward.

- For example, `121` is a palindrome while `123` is not.

Example 1:

```
Input: x = 121
Output: true
Explanation: 121 reads as 121 from left to right and from right to left.
```

121

123

```java
class Solution {

    public int reverse(int x) {
        int rev = 0;

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

        return rev;
    }

    public boolean isPalindrome(int x) {
        if(x < 0) return false;
        int reverse = reverse(x);
        if(reverse == x) return true;
        return false;
    }
}
```

Inverse Of A Number

Easy

Prev Next

1. You are given a number following certain constraints.
2. The key constraint is if the number is 5 digits long, it'll contain all the digits from 1 to 5 without missing any and without repeating any. e.g. 23415 is a 5 digit long number containing all digits from 1 to 5 without missing and repeating any digit from 1 to 5.Take a look at few other valid numbers - 624135, 81456273 etc.Here are a few invalid numbers - 139, 7421357 etc.
3. The inverse of a number is defined as the number created by interchanging the face value and index of digits of number.e.g. for 426135 (reading from right to left, 5 is in place 1, 3 is in place 2, 1 is in place 3, 6 is in place 4, 2 is in place 5 and 4 is in place 6), the inverse will be 416253 (reading from right to left, 3 is in place 1, 5 is in place 2,2 is in place 3, 6 is in place 4, 1 is in place 5 and 4 is in place 6) More examples - inverse of 2134 is 1243 and inverse of 24153 is 24153
4. Take as input number "n", assume that the number will follow constraints.
5. Print it's inverse.

Input Format

"n" where n is any integer, following constraints defined above.

$N \rightarrow 1$ to N digits

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

$\rightarrow$ place value

$\rightarrow$ face value

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

$\rightarrow$ Inverse

Number

$$\boxed{\sum fv * 10^{pv-1}}$$

4	3	2	1
1	2	7	8

$$\boxed{\sum pv * 10^{fv-1}} \rightarrow \text{Inverse}$$

$$8 * 10^0 + 7 * 10^1 + 2 * 10^2 + 1 * 10^3$$

pv = 78

iv = 73425681

fv = 4 3 8 2

$8 * 10^1$

← 80

```java
import java.util.*;

public class Main{

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

  int pv = 1;
  int inverse = 0;

  while(n > 0) {
      int fv = n % 10;
      n /= 10;

      inverse += pv * (int)Math.pow(10,fv-1);
      pv++;
  }

  System.out.println(inverse);

}
}
```

Getting Started - lecture 6

(Pattern Printing)

Concepts ↓

1. Pattern Printing GfG
2. Right Angled Triangle
3. Patternify
4. Right Angled Triangle - 2
5. Yet Another Pattern
6. Diamond of Stars
7. Void of Diamond
8. Backward Slash
9. forward Slash
10. Cross of Stars

Pattern - Printing (GfG)

Pattern Printing

School Accuracy: **65.5%** Submissions: **2776** Points: **0**

Given a number **N**. The task is to print a series of asterisk(*****) from **1 till N** terms with increasing order and difference being 1.

Example 1:

Input:

N = 3

Output:

* ** ***

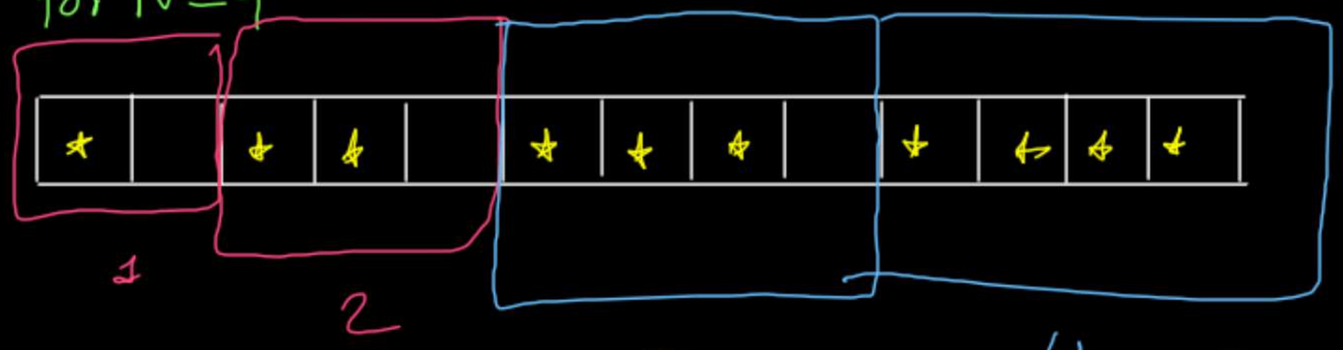
Explanation:

First, print 1 asterisk then space after that print 2 asterisk and space after that print 3 asterisk now stop as N is 3.

for N=3

Task → Print *
Print space

for N=4



↳ ith iteration → istars

```java
//User function Template for Java
class Solution{
    static void printPattern(int n){
        // code here

        for(int i=1;i<=n;i++) {

            for(int j=1;j<=i;j++) {
                System.out.print("*");
            }

            System.out.print(" ");
        }
    }
}
```

1 3

2

Nested Loop → One or many loops inside another

N = 3

i: ~~1~~ ~~2~~ ~~3~~ 4

j: ~~1~~ ~~2~~ ~~3~~ 4

* _ ** _ *** _

Right Angled Triangle

Pattern 1

Easy

Prev Next

1. You are given a number n.
2. You've to create a pattern of * and separated by tab as shown in output format.

Input Format

A number n

Output Format

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

for N=5

	1	2	3	4	5
1	*				
2	*	*			
3	*	*	*		
4	*	*	*	*	
5	*	*	*	*	*

for N=3

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

for N=3

*		*	*		*	*	*

for N=6

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

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

    // write ur code here
    int n = scn.nextInt();

     for(int i=1;i<=n;i++) {

        for(int j=1;j<=i;j++) {
            System.out.print("*\t");
        }

        System.out.println();
    }

}
```

Patternify

Pattern 2

Easy

Prev Next

1. You are given a number n.
2. You've to create a pattern of * and separated by tab as shown in output format.

Input Format

A number n

Output Format

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

for N = 5

	1	2	3	4	5
1	*	*	*	*	*
2	*	*	*	*	
3	*	*	*		
4	*	*			
5	*				

N to 1 → loop

↓

ith iteration → i stars

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

    // write ur code here
    int n = scn.nextInt();

    for(int i=n;i>=1;i--) {

        //stars
        for(int j=1;j<=i;j++) {
            System.out.print("*\t");
        }

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

1 to N Loop with External Variable

1 to N → Loop

for N=5 Stars 5 4 3 2 1

0

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

    // write ur code here
    int n = scn.nextInt();

    int stars = n;
    for(int i=1;i<=n;i++) {

        //stars
        for(int j=1;j<=stars;j++) {
            System.out.print("*\t");
        }

        System.out.println();
        stars--;
    }
}
```

3rd 4th 5th 6th

* * * * *
* * * *
* * *
* *
*

1 to N Loop Without External Variable

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

    // write ur code here
    int n = scn.nextInt();

    for(int i=1;i<=n;i++) {

        //stars
        for(int j=1;j<=n-i+1;j++) {
            System.out.print("*\t");
        }

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

Right Angled Triangle -2

Pattern 3

Easy

Prev Next

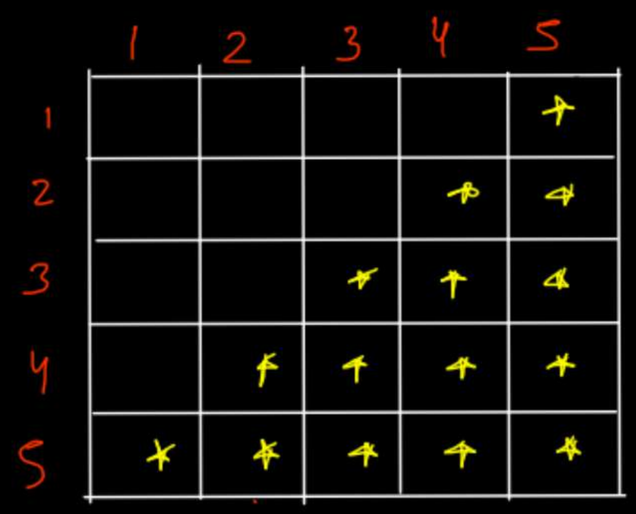
1. You are given a number n.
2. You've to create a pattern of * and separated by tab as shown in output format.

Input Format

A number n

Output Format

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



(5)

1st $N-i = 4$

2nd $N-i = 3$

3rd $N-i = 2$

4th $N-i = 1$

5th $N-i = 0$

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

    // write ur code here
    int n = scn.nextInt();

    for(int i=1;i<=n;i++) {
        //space
        for(int sp=1;sp<=n-i;sp++) {
            System.out.print("\t");
        }

        //star
        for(int j=1;j<=i;j++) {
            System.out.print("*\t");
        }

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

Yet another Pattern

11 Difficulty: EASY

Avg. time to solve
15 min

Success Rate
85%

Problem Statement

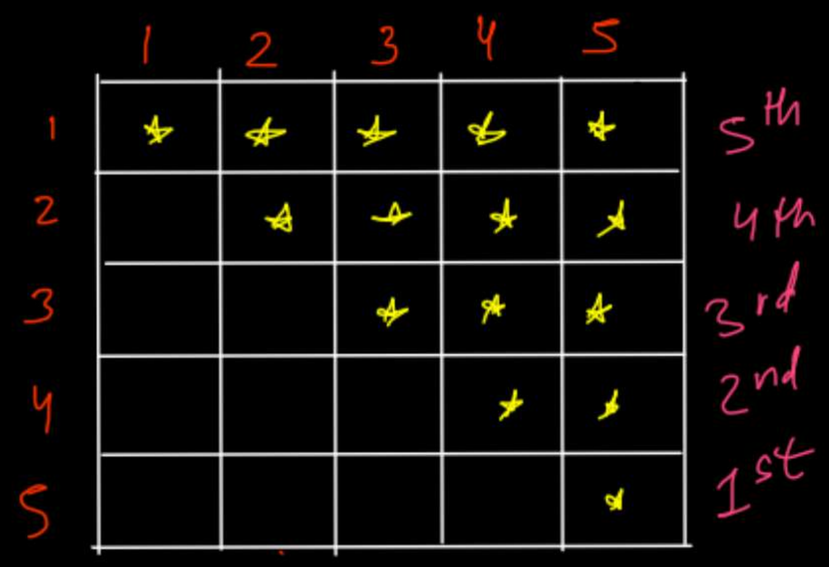
Suggest Edit

Ninja was playing with her sister to solve a puzzle pattern. However, even after taking several hours, they could not solve the problem.

A value of N is given to them, and they are asked to solve the problem. Since they are stuck for a while, they ask you to solve the problem. Can you help solve this problem?

```
Example : Pattern for N = 4

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

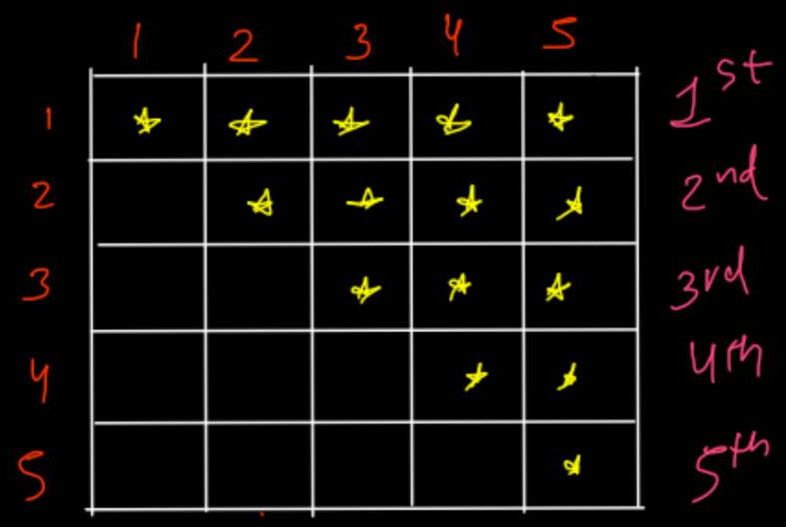
N to 1 Loop with external variable for space




```java
// Write your code here.

int space = 0;

for(int i=n;i>=1;i--) {

    for(int sp=1;sp<=space;sp++) {
        System.out.print(" ");
    }

    for(int j=1;j<=i;j++) {
        System.out.print("*");
    }

    space++;
    System.out.println();
}
}
```

1 2 3 4 5

0



$(N-i+1)$ Stars

Spaces → $i-1$

```java
// Write your code here.

for(int i=1;i<=n;i++) {
    //space

    for(int sp=1;sp<=i-1;sp++) {
        System.out.print(" ");
    }

    //stars

    for(int j=1;j<=n-i+1;j++) {
        System.out.print("*");
    }

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

Diamond of Stars

17

Difficulty: EASY

Contributed By
Ankush Gupta | Level 1

Avg. time to solve
10 min

Success Rate
90%

Problem Statement

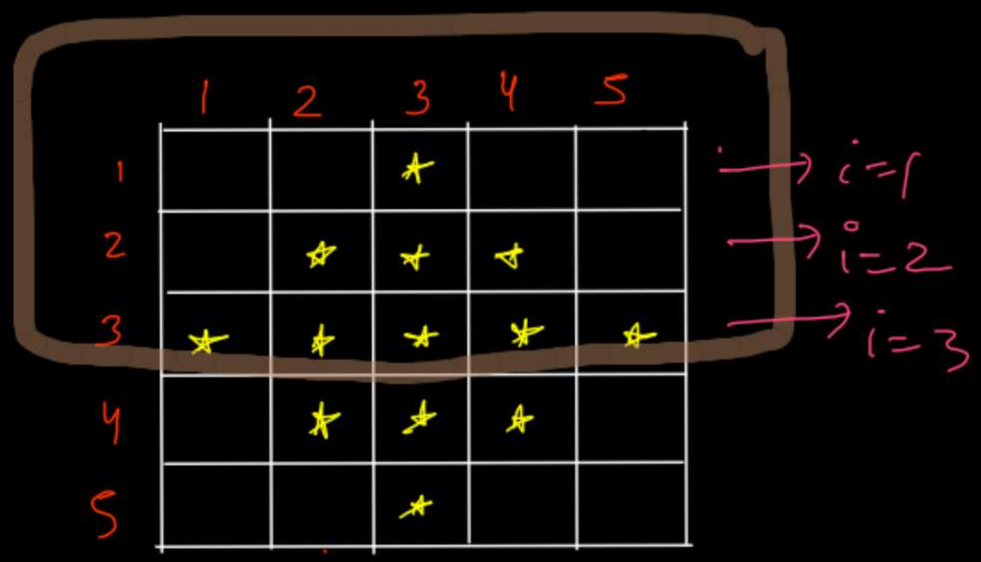
Suggest Edit

You are given an integer 'N'. Your task is to print the following pattern for the 'N' number of rows.

For Example:

```
Pattern for 'N' = 5:
```

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

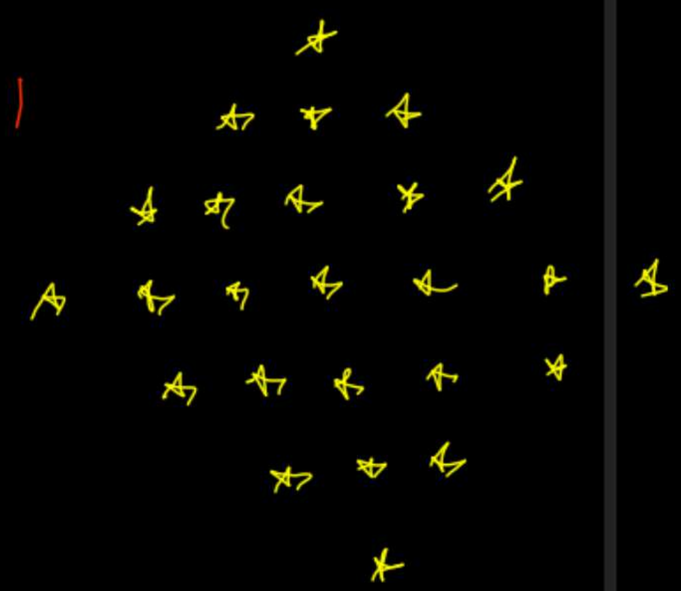


Upper half

Let $k = n/2 + 1$; $\rightarrow 3$

Spaces $\rightarrow k - i$

Stars $\rightarrow 2i - 1$



Second Half

Stars $\rightarrow$ start from $N-2$ & keep decrementing by 2

Spaces $= i$

```java
int k = n/2 + 1;

//first half
for(int i=1;i<=k;i++) {
    //spaces
    for(int sp=1;sp<=k-i;sp++) {
        System.out.print(" ");
    }

    //stars
    for(int j=1;j<=2*i-1;j++) {
        System.out.print("*");
    }

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

```java
//second half
int stars = n-2;
for(int i=1;i<=n-k;i++) {

    //spaces
    for(int sp=1;sp<=i;sp++) {
        System.out.print(" ");
    }

    //stars

    for(int j=1;j<=stars;j++) {
        System.out.print("*");
    }

    System.out.println();
    stars -= 2;
}
```

Backward Slash

Pattern 7

Easy

Prev Next

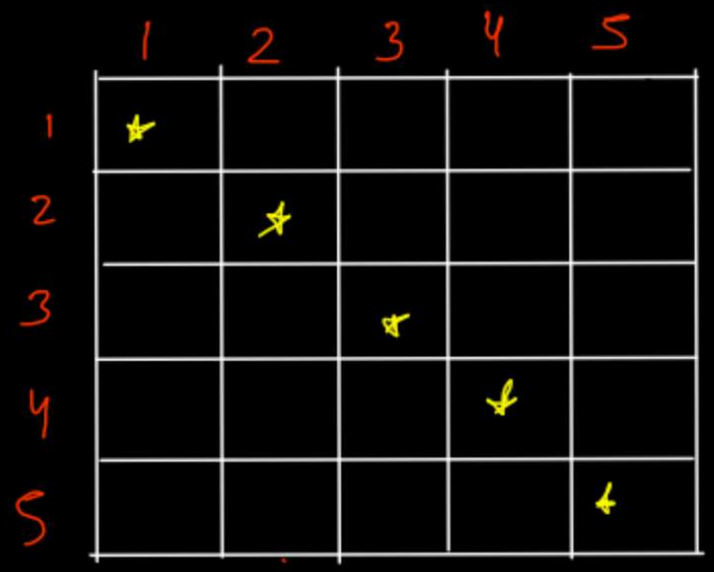
1. You are given a number n.
2. You've to create a pattern of * and separated by tab as shown in output format.

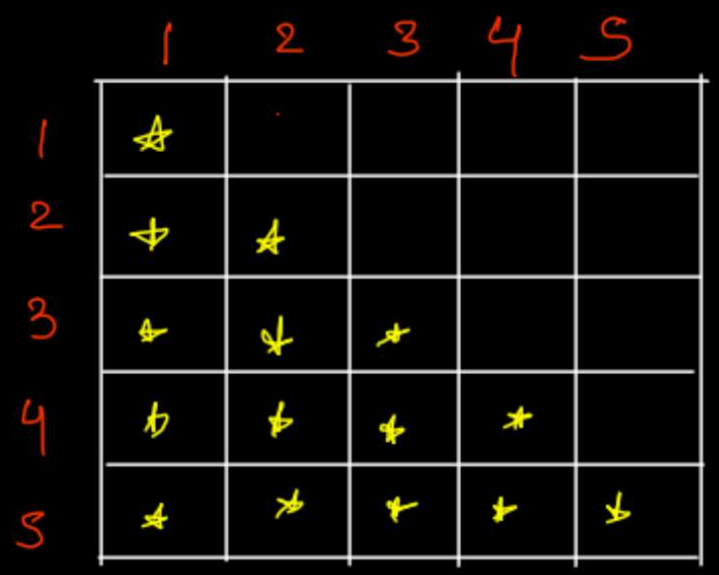
Input Format

A number n

Output Format

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





Compare them to simplify

1st
2nd
3rd
4th
5th

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

    // write ur code here

    int n = scn.nextInt();

    for(int i=1;i<=n;i++) {

        for(int j=1;j<=i;j++) {
            if(i == j) {
                System.out.print("*\t");
            } else {
                System.out.print("\t");
            }
        }

        System.out.println();
    }

}
```

forward Slash

Pattern 8

Easy

Prev Next

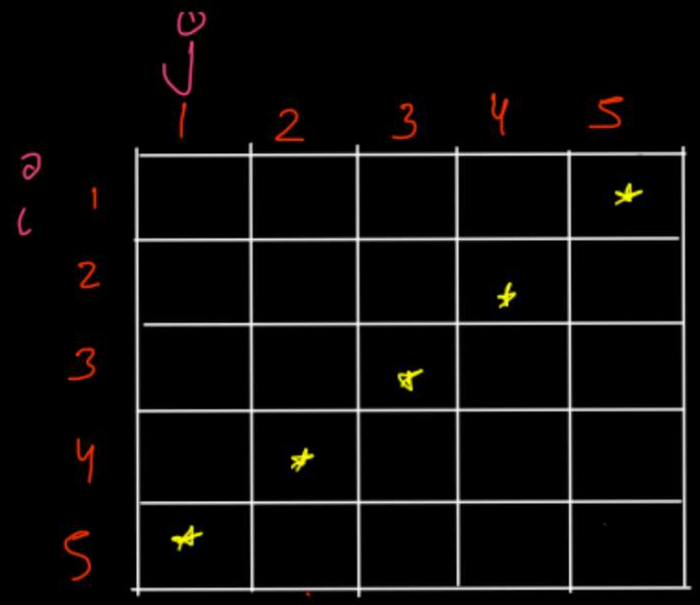
1. You are given a number n.
2. You've to create a pattern of * and separated by tab as shown in output format.

Input Format

A number n

Output Format

```
*
*
*
*
```



$(i+j == N+1) \rightarrow$ * break

else $\rightarrow$ Space

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

    // write ur code here
    int n = scn.nextInt();

    for(int i=1;i<=n;i++) {
        for(int j=1;j<=n;j++) {
            if(i+j == n+1) {
                System.out.print("*\t");
                break;
            } else {
                System.out.print("\t");
            }
        }
        System.out.println();
    }
}
}
```

Cross of Stars

Pattern 9

Easy

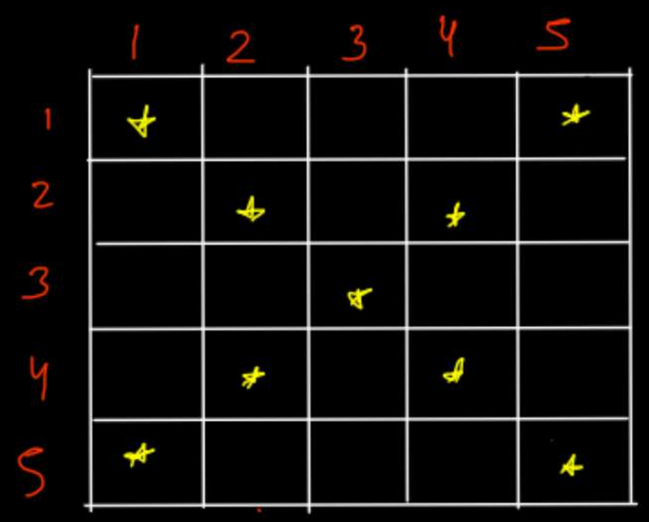
Prev Next

1. You are given a number n.
2. You've to create a pattern of * and separated by tab as shown in output format.

Input Format

A number n

Output Format




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

    // write ur code here
    int n = scn.nextInt();

    for(int i=1;i<=n;i++) {
        for(int j=1;j<=n;j++) {
            if(i == j || i+j == n+1) {
                System.out.print("*\t");
            } else {
                System.out.print("\t");
            }
        }

        System.out.println();
    }

}
```

Void of Diamond.

Pattern 6

Easy

Prev Next

1. You are given a number n.
2. You've to create a pattern of * and separated by tab as shown in output format.

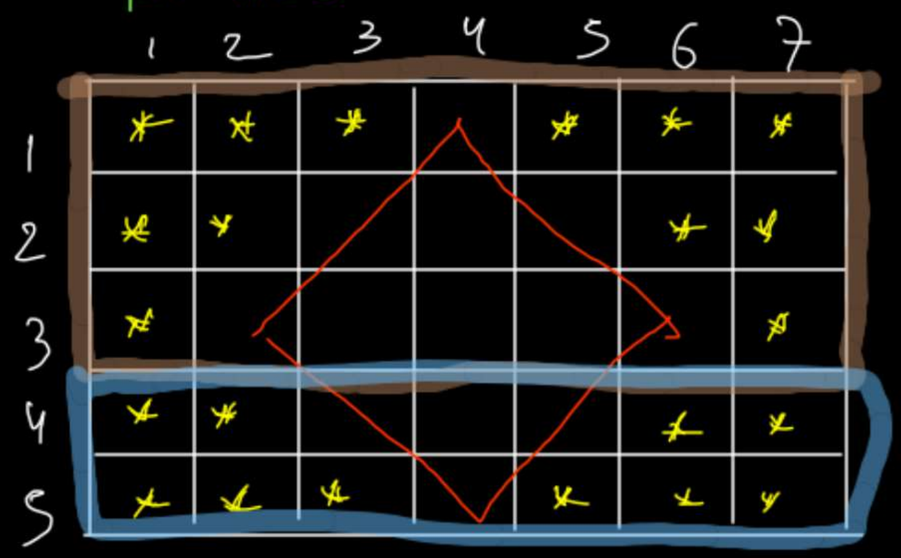
Input Format

A number n

Output Format

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

for n=5

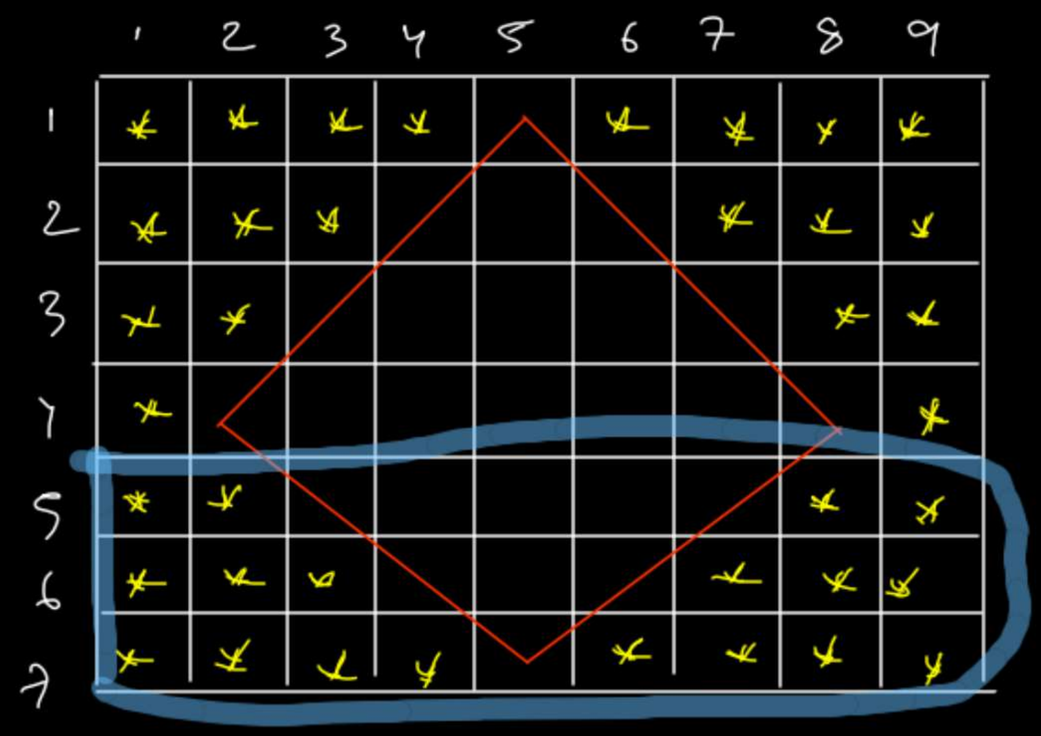


$k = n/2 + 1$

$\downarrow$

3

for n=7



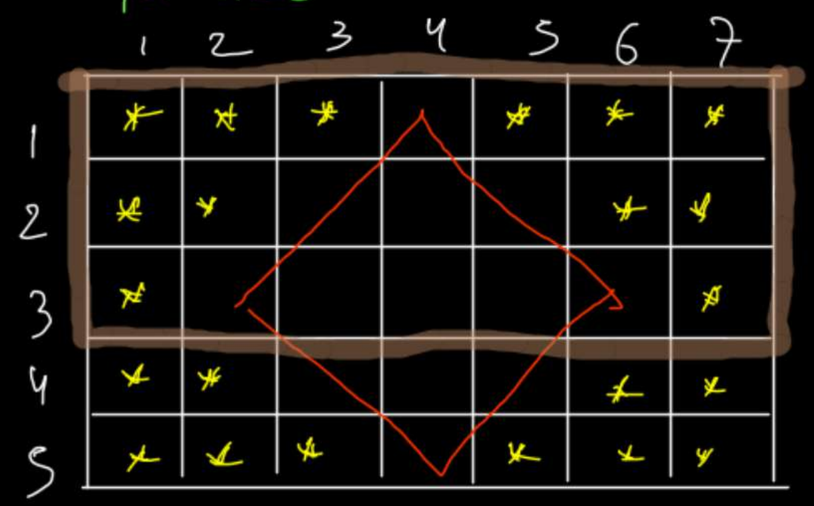
$k = 4$

first Half

→ Stars → Start from k

& dec by 1

for n=5

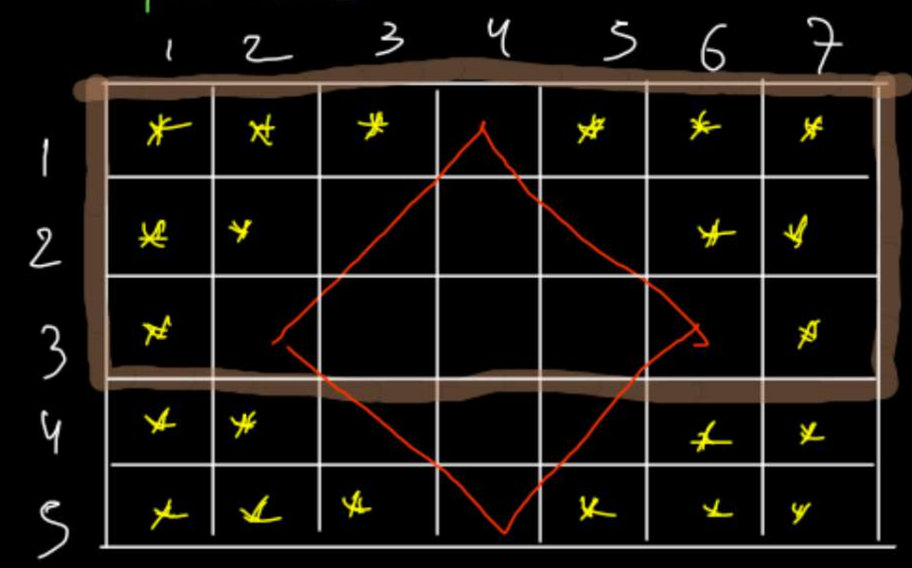


Second Half

Stars → start from 2
& inc by 1

Spaces → Start from n-2
& dec by 2

for n=5



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

    // write ur code here
    int n = scn.nextInt();

    int k= n/2 + 1;

    int stars = k;
    for(int i=1;i<=k;i++) {

        for(int j=1;j<=stars;j++) {
            System.out.print("*\t");
        }

        //spaces

        for(int sp=1;sp<=2*i-1;sp++) {
            System.out.print("\t");
        }

        for(int j=1;j<=stars;j++) {
            System.out.print("*\t");
        }

        System.out.println();
        stars--;
    }
```

```java
    stars = 2;
    int space = n-2;

    for(int i=1;i<=n-k;i++) {
        //stars
        for(int j=1;j<=stars;j++) {
            System.out.print("*\t");
        }

        for(int sp=1;sp<=space;sp++) {
            System.out.print("\t");
        }


        for(int j=1;j<=stars;j++) {
            System.out.print("*\t");
        }

        System.out.println();
        stars++;
        space -=2;
    }

}
```

Getting Started Lecture : 7

1. Number Pattern → Pep, CN
2. Cross of Numbers
3. Empty Diamond
4. Fibonacci Triangle
5. Mirror image of Triangle

Number Pattern

Pattern 11

Easy

Prev Next

1. You are given a number n.
2. You've to create a pattern as shown in output format.

Input Format

A number n

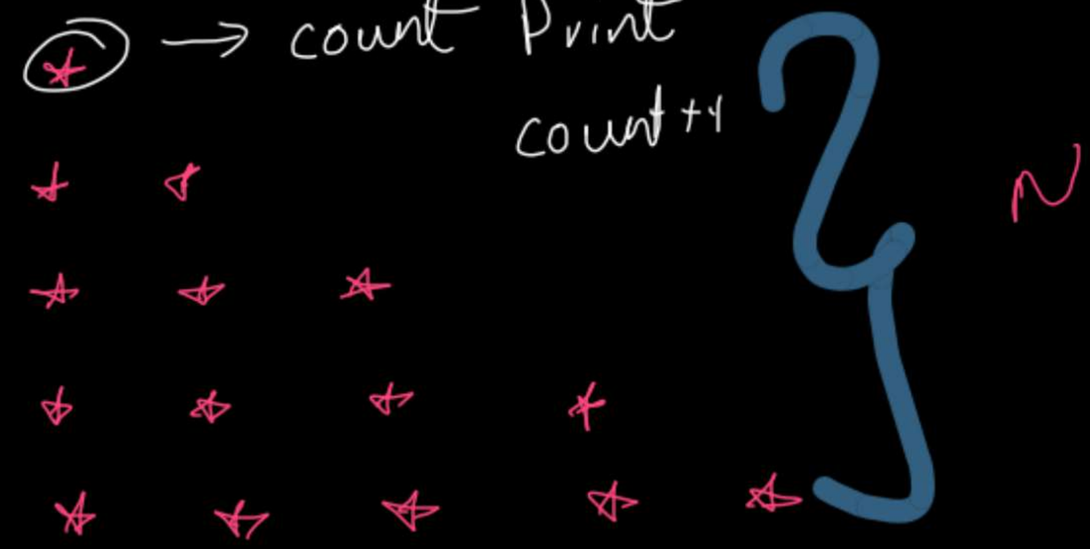
Output Format

```
1
2   3
4   5   6
7   8   9   10
..
```

N=5

	1	2	3	4	5
1	1				
2	2	3			
3	4	5	6		
4	7	8	9	10	
5	11	12	13	14	15

count = 1



N=5

	1	2	3	4	5
1	1				
2	2	3			
3	4	5	6		
4	7	8	9	10	
5	11	12	13	14	15

1 →

2 — 3 —

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

    // write ur code here
    int n = scn.nextInt();

    int num = 1;

    for(int i=1;i<=n;i++) {
        //numbers

        for(int j=1;j<=i;j++) {
            System.out.print(num + "\t");
            num++;
        }

        //leave line
        System.out.println();
    }

}
```

① ③ ②

N=5

num: ~~1~~ ~~2~~ ~~3~~ 4

i: ~~1~~ ~~2~~ 3

j: ~~1~~ ~~2~~ ③

Problem Statement

Ninja wants to build a number pattern for the given integer.

```
For example, If the given integer 'N' is 4

Pattern:

1
23
345
4567
```

N=5

	1	2	3	4	5	
1	1					1st
2	2	3				2nd
3	3	4	5			3rd
4	4	5	6	7		4th
5	5	6	7	8	9	5th

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


        for(int i=1;i<=n;i++) {
            //numbers
            int num = i;
            for(int j=1;j<=i;j++) {
                System.out.print(num + "\t");
                num++;
            }

            //leave line
            System.out.println();
        }
    }
}
```

→ This code is for Number Pattern on CN (Print only)

Cross of Numbers

Play with Pattern

7 Difficulty: EASY

Avg. time to solve
20 min

Success Rate
80%

Problem Statement Suggest Edit

Ninja wanted to go to a party along with his friends. However, his mom wanted him to go only if he completes a task assigned by her.

She gave Ninja a value and asked him to print an X-shaped pattern.

Example : Pattern for N = 3 (No. of rows = 5, No. of columns = 5) :

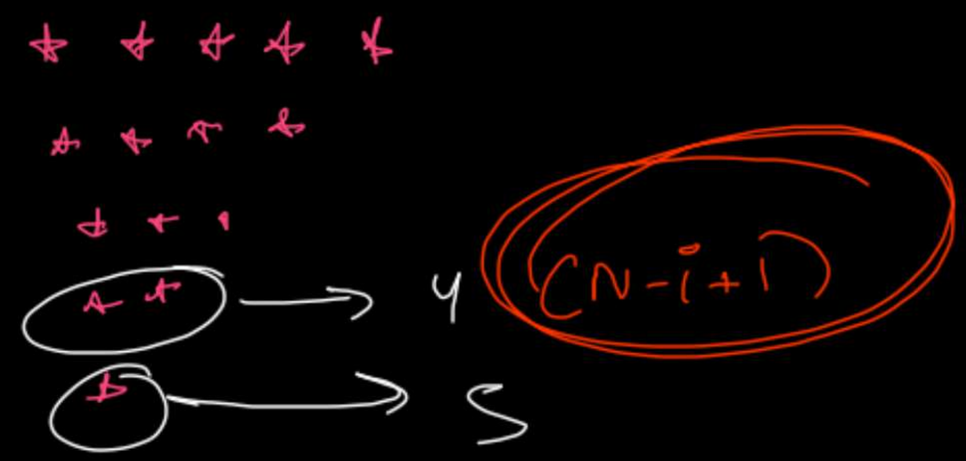
```
1   1
 2 2
  3
 2 2
1   1
```

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

$(5-4+1)=2$

$(5-5+1)=1$

$i \quad j \quad N$



```java
public static void ninjaPattern(int N) {

    // Write your code here.
    // N= 3
    // n = 2*N-1

    int n = 2*N-1;

    for(int i=1;i<=n;i++) {
        for(int j=1;j<=n;j++) {
            if(i == j || i+j == n+1) {
                if(i <= n/2 + 1) System.out.print(i);
                else System.out.print(n-i+1);
            } else {
                System.out.print(" ");
            }
        }
        System.out.println();
    }

}
```

Empty Diamond

Pattern 10

Easy

Prev Next

1. You are given a number n.
2. You've to create a pattern of * and separated by tab as shown in output format.

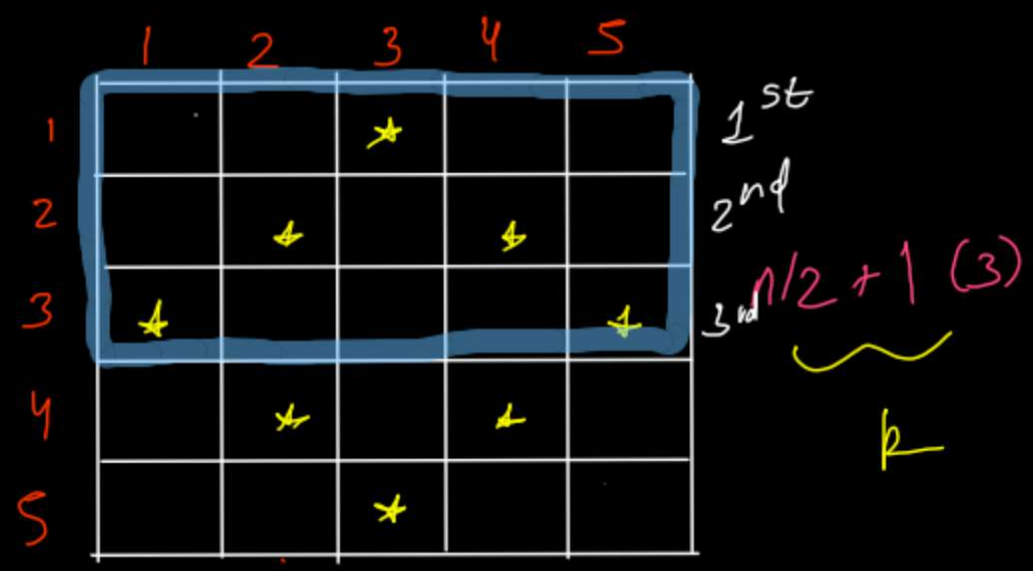
Input Format

A number n

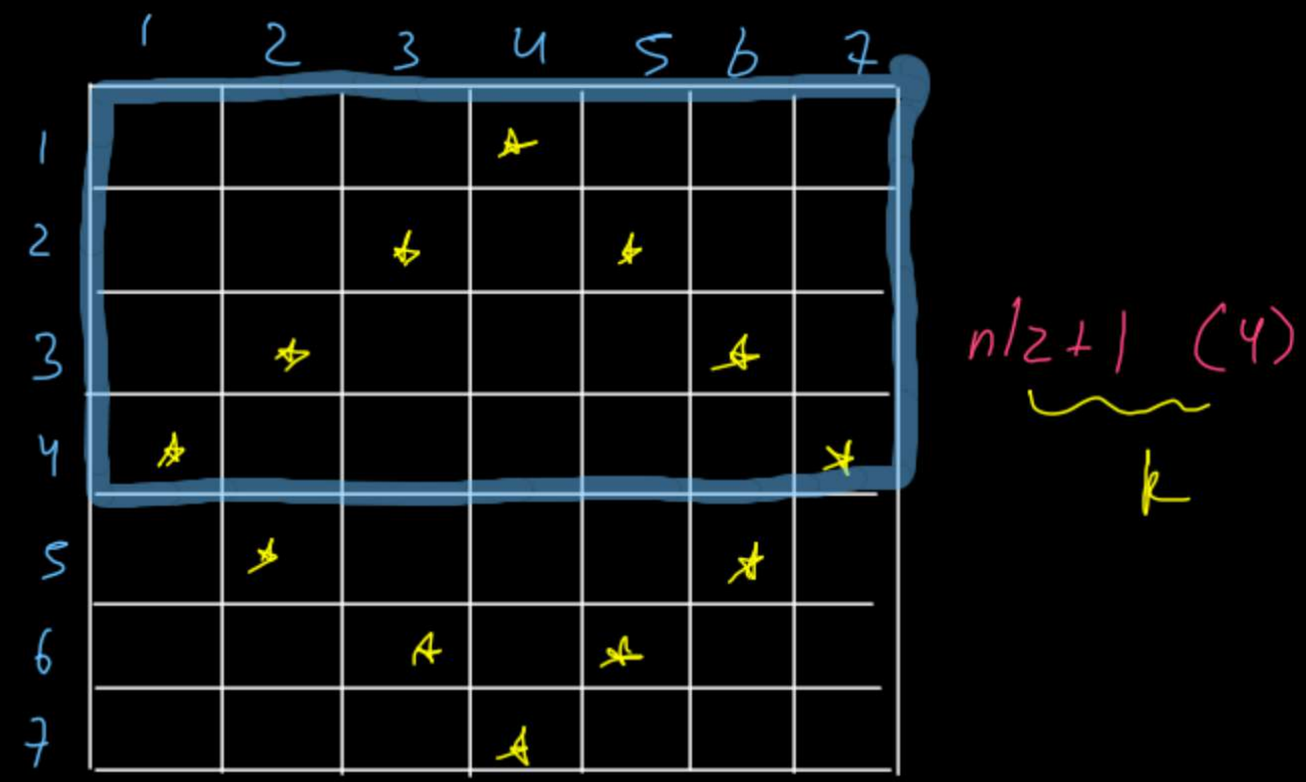
Output Format

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

for N = 5



N = 7



first

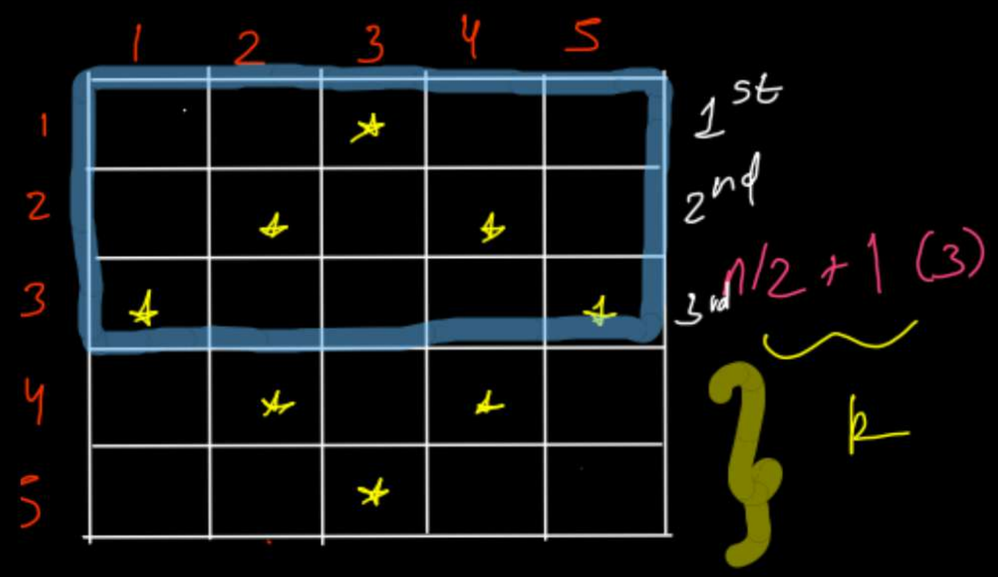
① k-i spaces

② 1 star

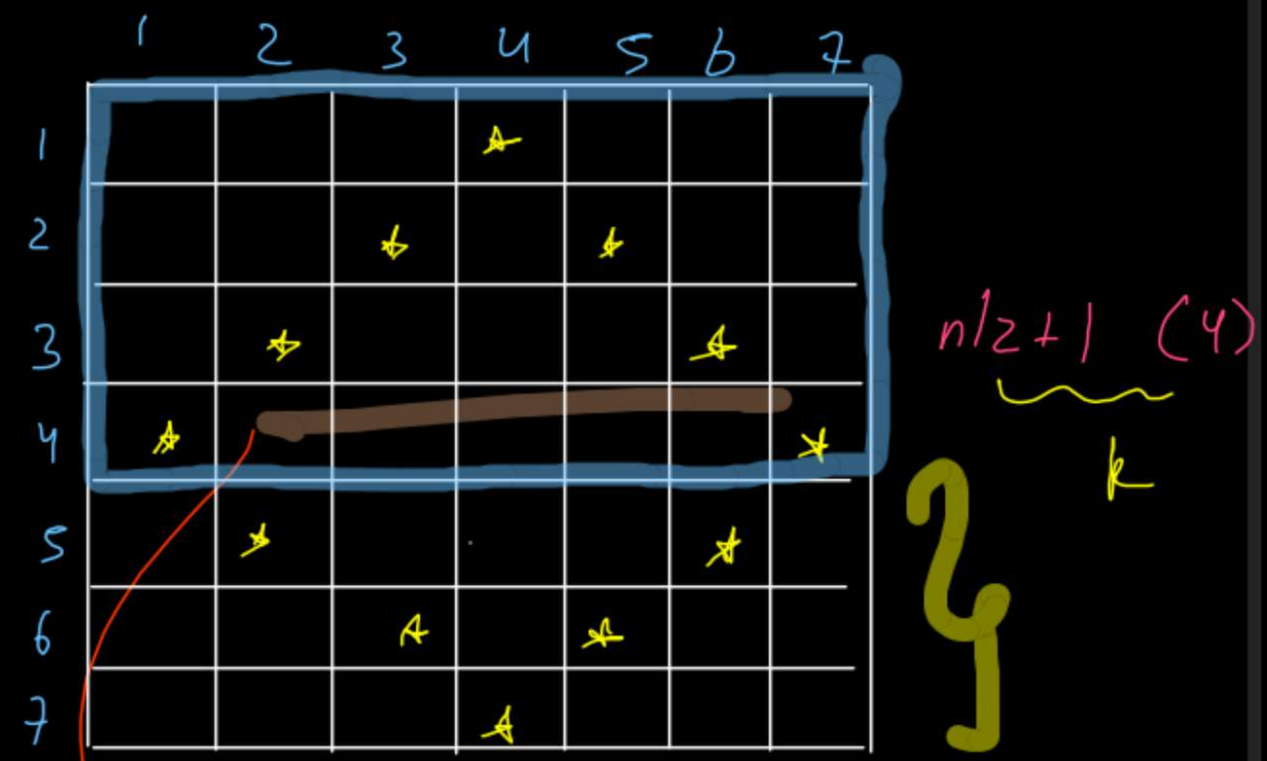
③ space starts from 1 & inc by 2 always

④ 1 star

for N=5



N=7



Second Half

① i spaces
② 1 star

③ space -= 4;
④ 1 star

last_space = space - 2;
sec_half_space = last_space - 2;

```
sp = 1
for(int = i; i<=3; i++) {
    print(space)
    sp+=2;
}
```

i = 1 2 3 4

1 3 5

space: 1 3 5 7

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

     // write ur code here
     int n = scn.nextInt();
    int space = 1;
     int k = n/2 + 1;

     for(int i=1;i<=k;i++) {
         for(int sp=1;sp<=k-i;sp++) {
             System.out.print("\t");
         }

         System.out.print("*\t");
         if(i == 1) {
             System.out.println();
             continue;
         }

         for(int sp=1;sp<=space;sp++) {
             System.out.print("\t");
         }

         System.out.println("*");
         space += 2;
     }
```

```java
    space -= 4; //this is explained in the class

    for(int i=1;i<=n-k;i++) {
        for(int sp=1;sp<=i;sp++) {
            System.out.print("\t");
        }

        System.out.print("*\t");
        if(i == n-k) {
            System.out.println();
            continue;
        }

        for(int sp=1;sp<=space;sp++) {
            System.out.print("\t");
        }

        System.out.print("*\t");

        space -=2;
        System.out.println();

    }
```

You are screen

Fibonacci Triangle

Pattern 12

Easy

Prev Next

1. You are given a number n.
2. You've to create a pattern as shown in output format.

Input Format

A number n

Output Format

```
0
1       1
2       3       5
8       13      21  34
..
```

	1	2	3	4	5
1	0				
2	1	1			
3	2	3	5		
4	8	13	21	34	
5	55	89	144	233	377

1st	2nd	3rd	4th	5th
0	1	1	2	3 .....

$$\boxed{fib(N) = fib(N-1) + fib(N-2)}$$

```java
// write ur code here
int n = scn.nextInt();
int a = 0;
int b = 1;
int c = 0;
for(int i=1;i<=n;i++) {
    for(int j=1;j<=i;j++) {
        if(i == 1) {
            System.out.print("0\t");
        } else if(i == 2 && j==1) {
            System.out.print("1\t");
        } else {
            c = a + b;
            System.out.print(c + "\t");
            a = b;
            b = c;
        }
    }
    System.out.println();
}
```

a = ~~0~~ ~~1~~ ~~1~~ ~~2~~ 3

b = ~~1~~ ~~1~~ ~~2~~ ~~3~~ 5

c = ~~0~~ ~~1~~ ~~2~~ ~~3~~ 5

$\boxed{\cancel{2}\,3}$ i

j = ~~1~~ ~~1~~ ~~2~~ ~~3~~ 4

0 —

→

1 — 1

→

2 — 3 — 5

Mirror image of triangle

3

Difficulty: EASY

Contributed By
Deepanshu_1780 | Level 1

Avg. time to solve
10 min

Success Rate
90%

Problem Statement

Suggest Edit

Ninja's younger sister got a project to print the pattern for the given number of rows, say 'N'. She finds it difficult to write such a pattern for every 'N'.

Ninja decided to help his sister and decided to write a code for this but wasn't able to write the code for the same. Ninja wants help to write a code and asks you for help. Help Ninja.

Example:

```
Pattern for N = 2
    0
   101
  21012
```

for N = 2

	1	2	3	4	5
0			0		
1		1	0	1	
2	2	1	0	1	2

{0 to N} → N+1 times

Spaces → N − i

2i+1 Numbers → Print

[1 to N] → N

1 2 3 4 . . . N

[1 - 5]

1 2 3 4 5

[0 to N-1] → N

[0 - 4]

0, 1, 2, 3, 4

```java
public class Solution {
    public static void NumberPattern(int n) {
        // Write your code here

        for(int i=0;i<=n;i++) {

            int num = i;

            for(int sp=1;sp<=n-i;sp++) {
                System.out.print(" ");
            }

            for(int j=1;j<=2*i+1;j++) {
                System.out.print(num);
                if(j < (2*i +1)/2 + 1) {
                    num--;
                } else {
                    num++;
                }
            }

            System.out.println();
        }

    }

}
```

$j \le 5$

for N=2

	1	2	3	4	5
0			0		
1		1	0	1	
2	2	1	0	1	2

num = ~~2~~ ~~1~~ ~~0~~ ~~1~~ 2

j = ~~1~~ ~~2~~ ~~3~~ ~~4~~ 5

$(2i+1)/2+1$

$=(2*2+1)/2+1$

$5/2+1$

$= 2+1 = 3$

2 1 0 1 2

Getting Started Lecture - 8

Pattern Printing

The Arrow Pattern

The Empty Hour Glass Pattern

Swastik

Pascals' Triangle **

Number System

Decimal to Any Base

Any base to Decimal

Any base to Any Base

Arrow Pattern

Pattern 17

Easy

Prev Next

1. You are given a number n.
2. You've to write code to print the pattern given in output format below.

Input Format

A number n

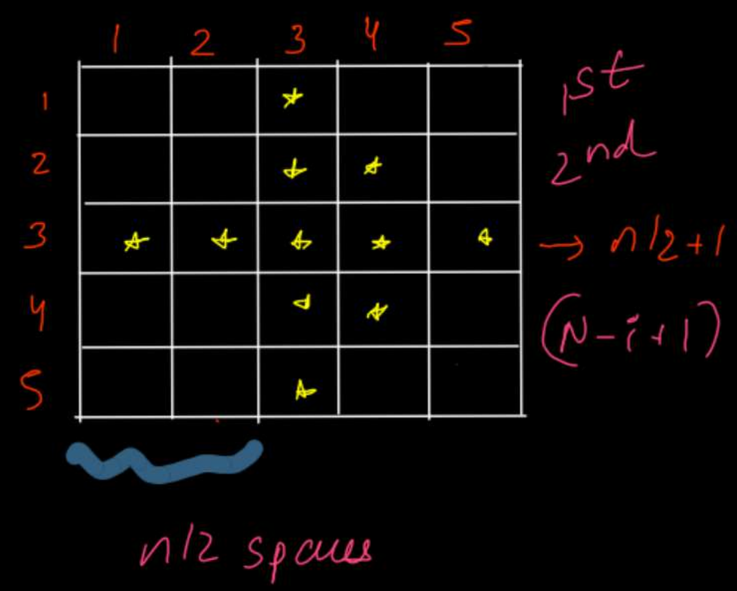
Output Format

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

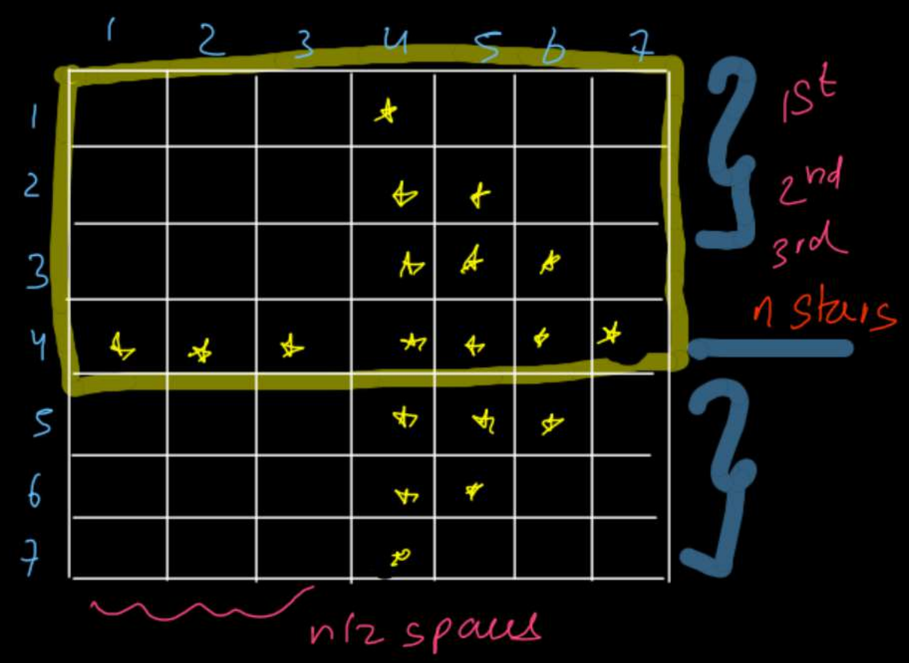
Upper Half → i stars # Space → always n/2

Lower Half → N-i+1

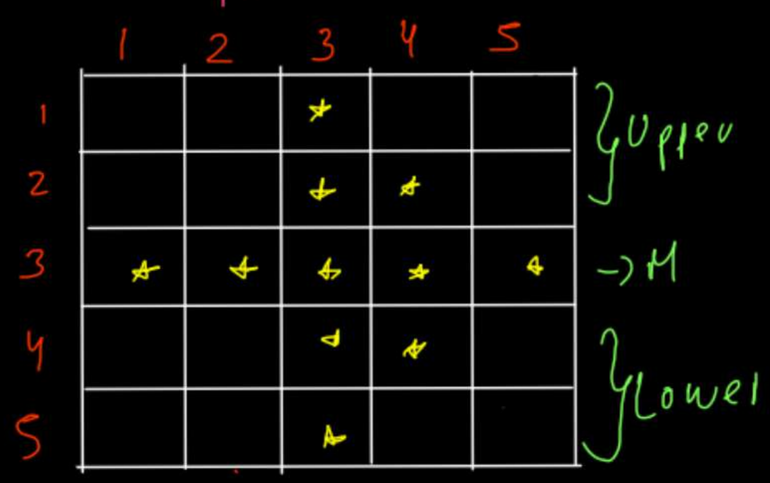
for N=5



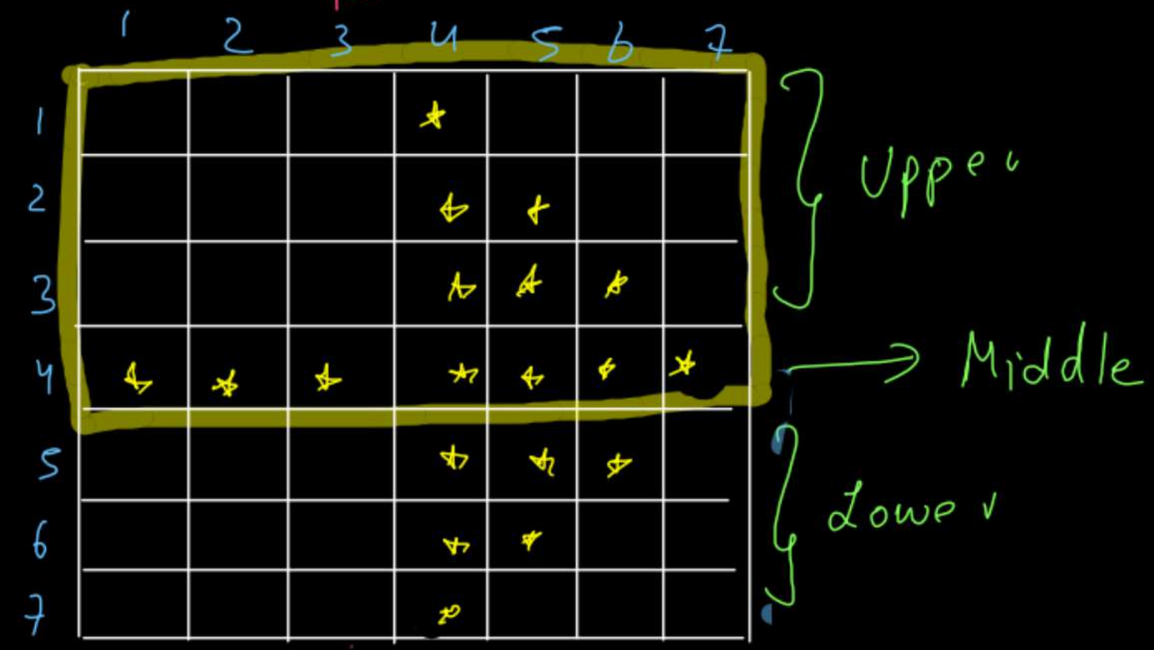
for N=7



for N=5



for N=7



Middle(n/2+1) → Print N Stars

Upper Half : i stars

Spaces : N/2

Lower Half : N-i+1

```java
import java.util.*;

public class Main {

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

        // write ur code here
        int n = scn.nextInt();

        for(int i=1;i<=n;i++) {
            if(i == n/2 + 1) {
                for(int j=1;j<=n;j++) {
                    System.out.print("*\t");
                }

            } else if(i <= n/2) {
                //upper half

                for(int sp=1;sp<=n/2;sp++) {
                    System.out.print("\t");
                }

                for(int j=1;j<=i;j++) {
                    System.out.print("*\t");
                }
            } else {
                for(int sp=1;sp<=n/2;sp++) {
                    System.out.print("\t");
                }

                for(int j=1;j<=n-i+1;j++) {
                    System.out.print("*\t");
                }
            }
            System.out.println();
        }

    }
}
```

The Empty Hour Glass Pattern

Pattern 18

Easy

Prev Next

1. You are given a number n.
2. You've to write code to print the pattern given in output format below

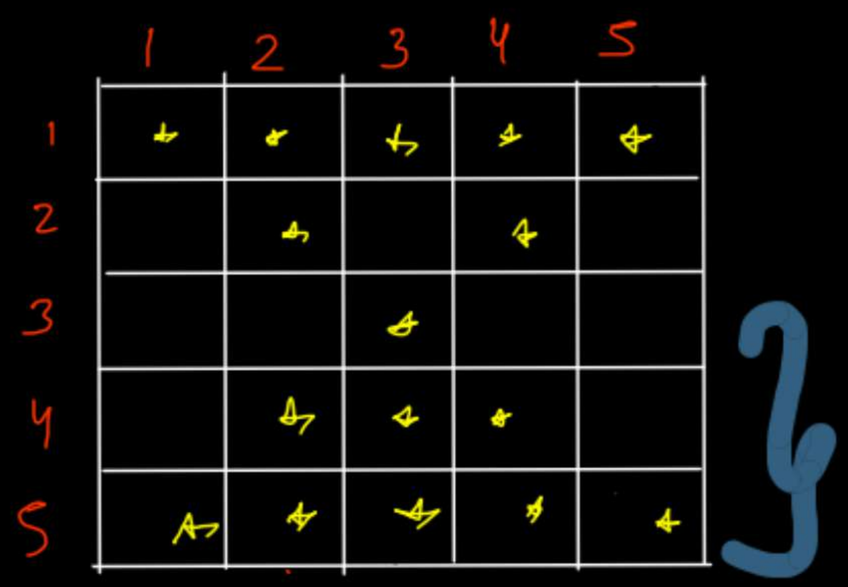
Input Format

A number n

Output Format

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

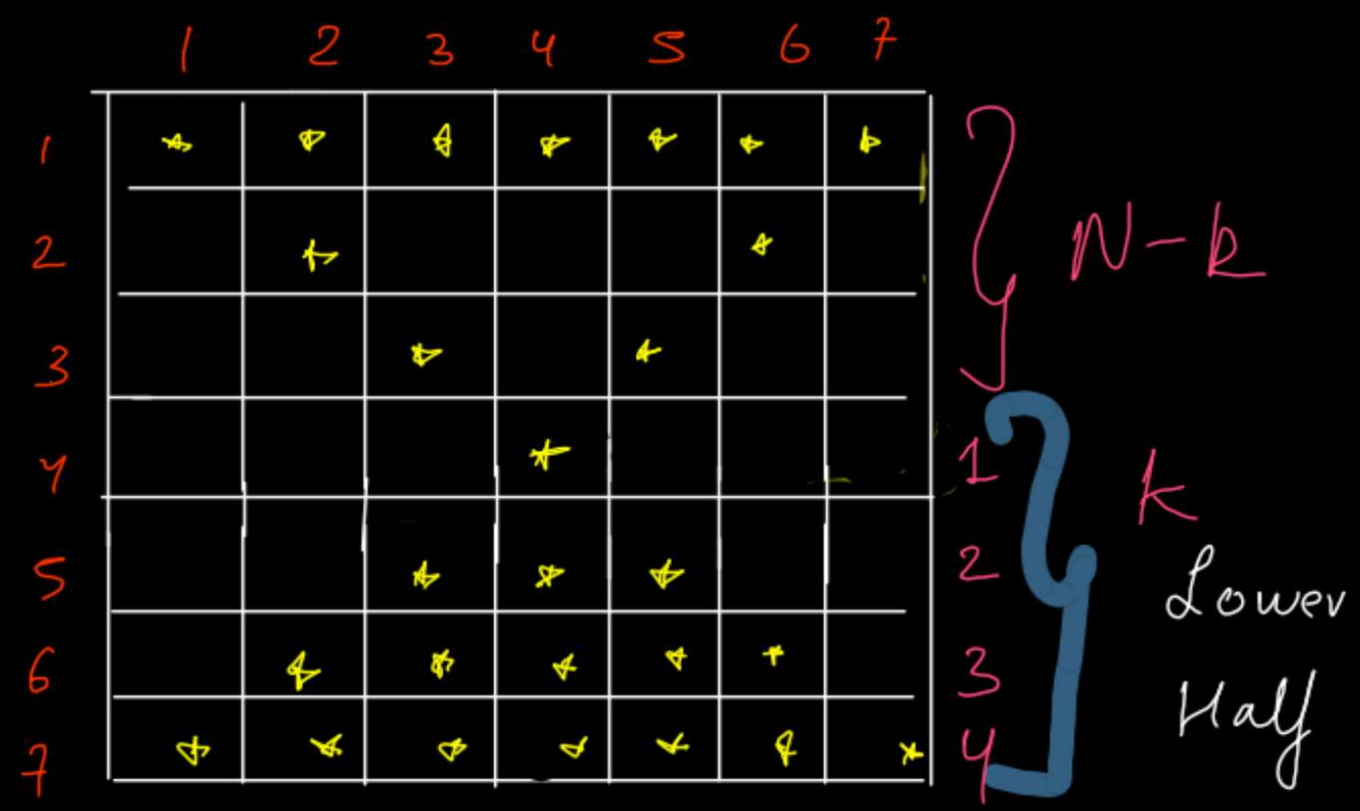
for $N = 5$



$k = n/2 + 1$

Upper Half $\rightarrow k - 1$

for $N = 7$



```java
        //upper half

        for(int i=1;i<=n-k;i++) {
            if(i == 1) {
                for(int j=1;j<=n;j++) {
                    System.out.print("*\t");
                }
            } else {
                for(int j=1;j<=n;j++) {
                    if(j == i || i+j == n+1) {
                        System.out.print("*\t");
                    } else {
                        System.out.print("\t");
                    }
                }
            }

            System.out.println();
        }

        //lower half

        for(int i=1;i<=k;i++) {
            for(int sp=1;sp<=k-i;sp++) {
                System.out.print("\t");
            }

            for(int j=1;j<=2*i-1;j++) {
                System.out.print("*\t");
            }

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

Swastik

Pattern 19

Easy

Prev Next

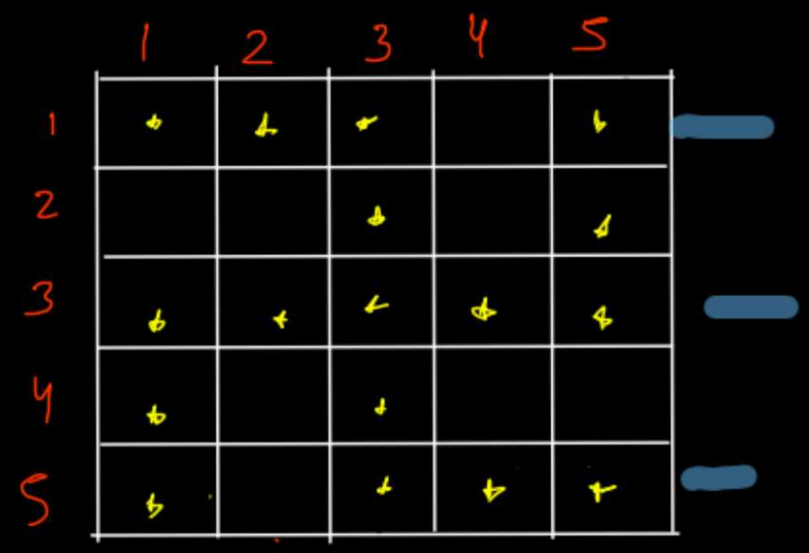
1. You are given a number n.
2. You've to write code to print the pattern given in output format below

Input Format

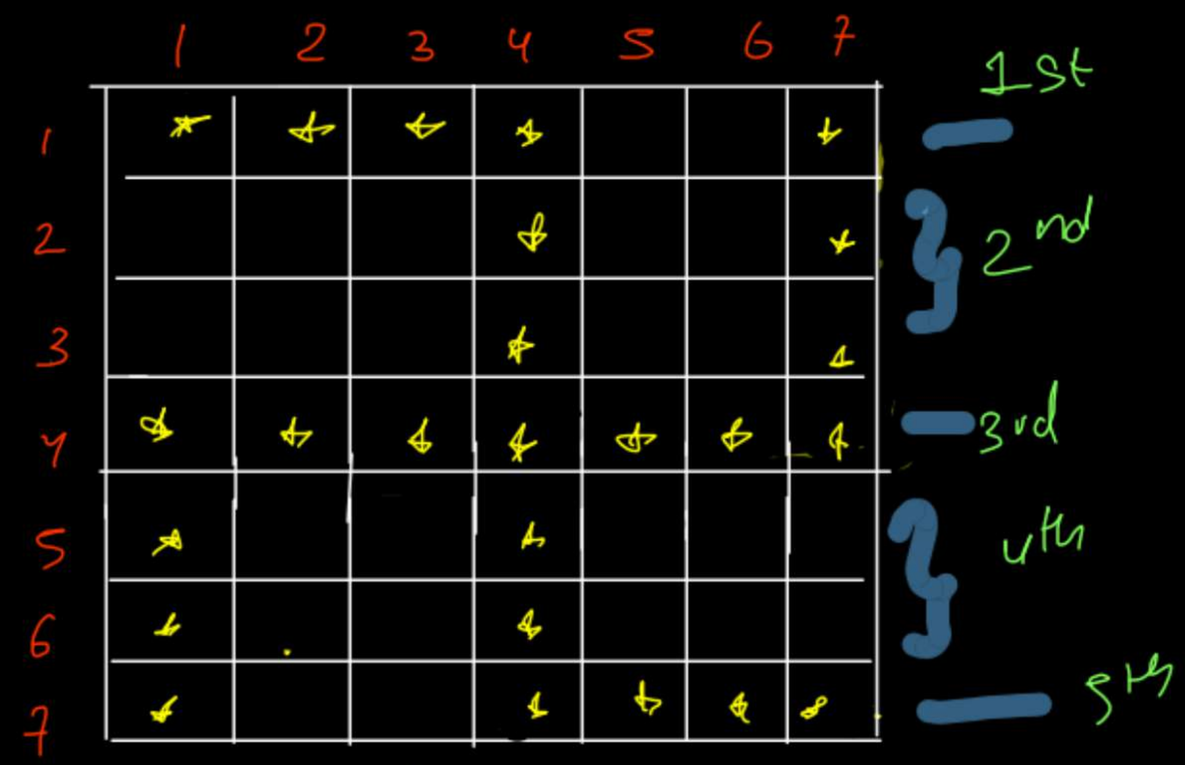
A number n

Output Format

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



for N=5



for N=7

1st Part

$(j \le n/2+1) \,||\, j==n \rightarrow$ Star

else Space

2nd

$j==n/2+1 \,||\, j==n \rightarrow$ Star

else $\rightarrow$ Space

3rd

Print N Stars

4th

$j == n/2+1 \;||\; j == 1 \rightarrow$ Stars

else $\rightarrow$ Space

5th

$j == 1 \;||\; j \geq n/2+1$

```java
for(int i=1;i<=n;i++) {
    if(i == 1) {
        for(int j=1;j<=n;j++) {
            if(j <= n/2 + 1 || j == n) {
                System.out.print("*\t");
            } else {
                System.out.print("\t");
            }
        }
    } else if(i <= n/2) {
        for(int j=1;j<=n;j++) {
            if(j == n/2 + 1 || j == n) {
                System.out.print("*\t");
            } else {
                System.out.print("\t");
            }
        }
    } else if(i == n/2 + 1) {
        for(int j=1;j<=n;j++) {
            System.out.print("*\t");
        }
    } else if(i <= n-1) {
        for(int j=1;j<=n;j++) {
            if(j == n/2 + 1 || j == 1) {
                System.out.print("*\t");
            } else {
                System.out.print("\t");
            }
        }
    } else {
        for(int j=1;j<=n;j++) {
            if(j >= n/2 + 1 || j == 1) {
                System.out.print("*\t");
            } else {
                System.out.print("\t");
            }
        }
    }

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

Pascals' Triangle

Pattern 13

Easy

Prev Next

1. You are given a number n.
2. You've to create a pattern as shown in output format

Input Format

A number n

Output Format

```
1
1       1
1       2       1
1       3       3       1
1       4       6       4       1
1       5       10      10      5       1
..
```

for N = 5

	0	1	2	3	4
0	1				
1	1	1			
2	1	2	1		
3	1	3	3	1	
4	1	4	6	4	1

$$^nC_k = {}^nC_{N-k}$$

$$^nC_0 = {}^nC_n = 1$$

$$^5C_1 = \frac{5!}{4!\,1!} = \frac{5 \times \cancel{4!}}{\cancel{4!}\,1!} = 5$$

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

$^4C_1$

for N = 5

	0	1	2	3	4
0	1				
1	1	1			
2	1	2	1		
3	1	3	3	1	
4	1	4	6	4	1

1 to N → N

0 to N-1 → N

$$^iC_j = {}^4C_1$$

$$^{n}C_{r+1} = \frac{^{n}C_{r}\,(n-r)}{r+1}$$

$$N!_{0} \rightarrow N*(N-1)!_{0}$$

LHS $$\frac{n!_{0}}{(n-(r+1))!_{0}\,(r+1)!_{0}} = \boxed{\frac{n!_{0}}{((n-r)-1)!\,(r+1)!_{0}}}$$

RHS $$\frac{n!_{0}\,(n-r)}{(n-r)!_{0}\,r!_{0}\,(r+1)} = \frac{n!_{0}\,\cancel{(n-r)}}{\cancel{(n-r)}((n-r)-1)!_{0}\,(r+1)!_{0}} = \boxed{\frac{n!_{0}}{((n-r)-1)!_{0}\,(r+1)!_{0}}}$$

	0	1	2	3	4
0	1				
1	1	1			
2	1	2	1		
3	1	3	3	1	
4	1	4	6	4	1

$$^{4}C_{0} \rightarrow 1$$

$$^{n}C_{r+1} = \frac{^{n}C_{r}\,(n-r)}{r+1}$$

$$^{i}C_{j+1} = \frac{^{i}C_{j}\,(i-j)}{j+1}$$

```java
    //write your code here

    int n = scn.nextInt();
    for(int i=0;i<n;i++) {
        int iCj = 1;
        for(int j=0;j<=i;j++) {
            System.out.print(iCj + "\t");
            int iCjp1 = iCj * (i-j)/(j+1);
            iCj = iCjp1;
        }
        System.out.println();
    }
}
```

Decimal to Any Base

Decimal To Any Base

Easy

Prev Next

1. You are given a decimal number n.
2. You are given a base b.
3. You are required to convert the number n into its corresponding value in base b.

Decimal Number System → digits from 0 to 9

radix or base → 10

Binary Number System → digits from 0 to 1

radix or base → 2

Hexadecimal NS → digits from 0 to 9

A → 10

B → 11

C → 12

D → 13

E → 14

f → 15

radix or base → 16

N = 25 B = 2

2	25	
2	12	1
2	6	0
2	3	0
2	1	1
	0	1

ans = 0;

p = 0

while(n != 0) {

int rem = N % B;

N /= B;

ans += rem * 10^p

p++;

}

$(11001)_2$

10^4 10^3 10^2 10^1 10^0

N = 25 B = 2

ans = ~~0~~ 1 + 0 = 1 + 0 = 1

p = ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~5 = 1001

rem = ~~1~~ ~~0~~ ~~0~~ ~~1~~ 1 × 10^4

n = ~~12~~ ~~6~~ ~~3~~ ~~1~~ 0

```
10,000
 1001
-----
11001
```

```java
public static int getValueInBase(int n, int b){
    // write code here

    int ans = 0;
    int p = 0;
    while(n != 0) {
        int rem = n % b;
        n /= b;
        ans += rem * (int)Math.pow(10,p);
        p++;
    }

    return ans;
}
}
```

```java
public static int getValueInBase(int n, int b){
    // write code here

    int ans = 0;
    int p = 0;
    while(n != 0) {
        int rem = n % b;
        n /= b;
        ans += rem * (int)Math.pow(10,p);
        p++;
    }

    return ans;
}
}
```

N = 25

① N = ~~25~~ 12 p = ~~0~~ 1

rem = 1 ans = 0 + 1 = 1

$1 * 10^0 = 1 * 1 = 1$

② N = ~~12~~ 6 p = ~~1~~ 2

rem = 0 ans = 1

$0 * 10^1 = 0$

③ N = ~~6~~ 3 p = ~~2~~ 3

rem = ~~0~~ 0 ans = 1

$0 * 10^2 = 0$

④ N = 3 p = 3

rem = 1 ans = 1001

Any Base To Decimal

Easy

Prev Next

1. You are given a number n.
2. You are given a base b. n is a number on base b.
3. You are required to convert the number n into its corresponding value in decimal number system.

$11001 \rightarrow N \qquad B = 2$

$$98634$$

$$10^4 \quad 10^3 \quad 10^2 \quad 10^1 \quad 10^0$$

$$11001$$

$$2^4 \quad 2^3 \quad 2^2 \quad 2^1 \quad 2^0$$

$$= 16 + 8 + 1$$

$$= 25$$

11001

ans += digit * base^{p}

```java
public static int getValueIndecimal(int n, int b){
    // write your code here

    int ans = 0;
    int pv = 0;
    while(n != 0) {
        int fv = n % 10;
        n /= 10;
        ans += fv * (int)Math.pow(b,pv);
        pv++;
    }

    return ans;
}
}
```

fv = 1

ans = 9 + 16 = 25

pv = ~~0~~ ~~1~~ ~~2~~ ~~3~~ 4

$1 + 2^4 = 16$

Any Base to Any Base

Any Base To Any Base

Easy

Prev Next

1. You are given a number n.
2. You are given a base b1. n is a number on base b.
3. You are given another base b2.
4. You are required to convert the number n of base b1 to a number in base b2.

3 → source Base

Any Base to Decimal

10

Decimal to Any Base

5 → destination Base

```java
public static int anyBaseToAnyBase(int n, int src,int dest) {
    int decimal = anyBaseToDecimal(n,src);
    int ans = decimalToAnyBase(decimal,dest);
    return ans;
}
```

Getting Started – Lecture 9

Number System

- → Any Base Addition
- → Any Base Subtraction
- → Any Base Multiplication

Any Base Addition

Any Base Addition

Easy

1. You are given a base b. 2. You are given two numbers n1 and n2 of base b. 3. You are required to add the two numbes and print their value in base b.

Constraints

2 <= b <= 10 0 <= n1 <= 256 0 <= n2 <= 256

Format

Input

A base b A number n1 A number n2

Output

A number representing the sum of n1 and n2 in base b.

$n_1 \rightarrow 1013$

$n_2 \rightarrow 99$

$b = 10$

$ca = 0$ $d \rightarrow 1+0+0$ $ca=0$

$ca \rightarrow$

```
 0 1 1
 1 0 1 3
 0 0 9 9
 -------
 1 1 1 2
```

$d_1 = n_1 \% 10;$ $n_1 /= 10$ (101)

$d_2 = n_2 \% 10;$ $n_2 /= 10$ (9)

```
if (d1+d2+ca < b) {
    d = d1+d2+ca;
    ca = 0;
} else {
    d = (d1+d2+ca)%b;
    ca = (d1+d2+ca)/b;
}
```

greater than or equal to base

while ($n_1 > 0$ || $n_2 > 0$ || $ca > 0$) → condition for addition

$$
\begin{array}{r}
1\ 1\ \ \\
1\ 0\ 9\ 9\ 9 \\
0\ \ \ \ \ \ 1 \\
\hline
1\ 0\ 0\ 0 \\
\hline
\end{array}
$$

101 3

99 9

$d_1 = n_1 \% 10$

$d_2 = n_2 \% 10$

$n_1 /= 10;$

$n_2 /= 10;$

$d = 2$

$ca = 1$

1023

10^3 10^2 10^1 10^0

1000100101

```
public static int getSum(int b, int n1, int n2){
    // write ur code here

    int ans = 0;
    int p = 1;
    int ca = 0;

    while(n1 > 0 || n2 > 0 || ca > 0) {
        int d1 = (n1 > 0) ? n1 % 10 : 0;
        int d2 = (n2 > 0) ? n2 % 10 : 0;
        n1 /= 10;
        n2 /= 10;

        int digit = 0;

        if(d1 + d2 + ca >= b) {
            digit = (d1 + d2 + ca) % b;
            ca = (d1 + d2 + ca)/b;
        } else {
             digit = d1 + d2 + ca;
             ca = 0;
        }

       ans += digit * p;
       p *= 10;

    }

    return ans;
}
}
```

Problem | Editorial | Submissions

Any Base Subtraction

Easy

1. You are given a base b.
2. You are given two numbers n1 and n2 of base b.
3. You are required to subtract n1 from n2 and print the value.

Constraints

```
2 <= b <= 10
0 <= n1 <= 256
n1 <= n2 <= 256
```

Format

Input

A base b
A number n1
A number n2

Output

A number of base b equal in value to n2 - n1.

$n_1 \rightarrow 99$

$n_2 \rightarrow 1012$

$1012 \rightarrow d_2$

$-99 \rightarrow d_1$

$n_1 \rightarrow 45$

$n_2 \rightarrow 99$

$99 \rightarrow d_2$

$-45 \rightarrow d_1$

54

while(n2>0) → condition for subtraction

$n_1 \rightarrow 99$

$n_2 \rightarrow 1012$

$b_o = 0$

$b = 10$

-1 -1 -1

$1012 \rightarrow n_2$

$0099 \rightarrow n_1$

0913

$d_2 = 1$

$d_1 = 0$

$b_o = -1$

$d_2 - d_1 + b_o$

$1 - 0 + (-1) = 0$

$b_o = 0$

```
if (d2-d1+bo < 0) {
    d = d2-d1+bo+base;
    bo = -1;
} else {
    d = d2-d1+bo;
    bo = 0;
}
```

$$\begin{array}{r} 1012 \\ 99 \\ \hline 0913 \end{array}$$

```java
public static int getDifference(int b, int n1, int n2){
    // write your code here
    int ans = 0;
    int p = 1;
    int bo = 0;

    while(n2 > 0) {
        int d2 = n2 % 10;
        int d1 = (n1 > 0) ? n1 % 10 : 0;

        n1 /= 10;
        n2 /= 10;

        int digit = 0;

        if(d2 - d1 + bo < 0) {
            digit = d2 - d1 + bo + b;
            bo = -1;
        } else {
            digit = d2 - d1 + bo;
            bo = 0;
        }

        ans += digit *p;
        p *= 10;
    }

    return ans;

}
```

Any Base Multiplication

Easy

1. You are given a base b.
2. You are given two numbers n1 and n2 of base b.
3. You are required to multiply n1 and n2 and print the value.

Constraints

```
2 <= b <= 10
0 <= n1 <= 10000
0 <= n2 <= 10000
```

Format

Input

A base b
A number n1
A number n2

Output

A number of base b equal in value to n2 * n1.

2023 → n1
* 15 → n2

Step 1

```
  1 1
  2023
     5
 -----
 10115
```

Step 2

```
  2023
     1
 -----
  2023   x 10
```

```
  10115
  20230
 -----
  30345
```

Step 3

Multiply No with digit in base B

① 0 1 1
2023 → n
5 → d

10115

ca = 0

$d_1 = 0$

$d = 5$

$0 * 5 + ① = ②$

$d_1 = n \% 10$

```
if (d1 * d + ca >= b) {
    d = (d1 * d + ca) % b;
    ca = (d1 * d + ca) / b;
}  else {
    d = d1 * d + ca;
}  ca = 0
```

Multiplication of Complete Nos in base B

$$\begin{array}{r} 2023 \\ \times\ 15 \\ \hline \end{array}$$

$$\underline{10115}$$

① Multiply No with Digit

$$\begin{array}{r} 2023 \\ \times\ 5 \\ \hline \end{array} \qquad \begin{array}{r} 2023 \\ 1 \\ \hline \end{array}$$

$$\underline{10115} \qquad \underline{20230}$$

8 8

②

$$\begin{array}{r} 10115 \\ 20230 \\ \hline \end{array}$$

```java
public static int multiplyWithDigit(int n, int d,int b) {
    int ans = 0;
    int p = 1;
    int ca = 0;

    while(n > 0 || ca > 0) {
        int d1 = n % 10;
        n /= 10;

        int digit = 0;
        if(d*d1 + ca >= b) {
            digit = (d*d1 + ca) % b;
            ca = (d*d1 + ca)/b;
        } else {
            digit = d*d1 + ca;
            ca = 0;
        }

        ans += digit*p;
        p*=10;
    }

    return ans;
}
```

```java
public static int getProduct(int b, int n1, int n2){
    // write your code here

    int ans = 0;
    int p = 1;

    while(n1 > 0) {
        int d1 = n1 % 10;
        n1 /= 10;

        int smallAns = multiplyWithDigit(n2,d1,b);
        ans  = anyBaseAddition(ans,smallAns * p,b);
        p *= 10;

    }

    return ans;
}
```

Getting Started – Lecture 9 (continued)

Arrays Intro

Memory Mapping

Array Traversal & Printing {HashCode & toString()}

Data Structure

Linear

Array

Linked List

Stack

Queue, Dequeue

Non-Linear

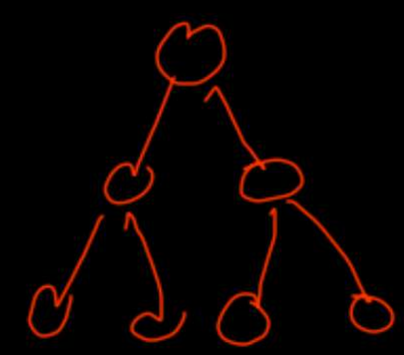
(Cyclic)

Circular

Circular LL

(Non-cyclic)

Non-circular



Array

```
int marks = 98;
int marks2 = 82;
int marks3 = 76;
 |
 |
 |
```

Contiguous collection of same datatype.

int x; → declaration

x = 10; → initialize

Java is a fool-proof language

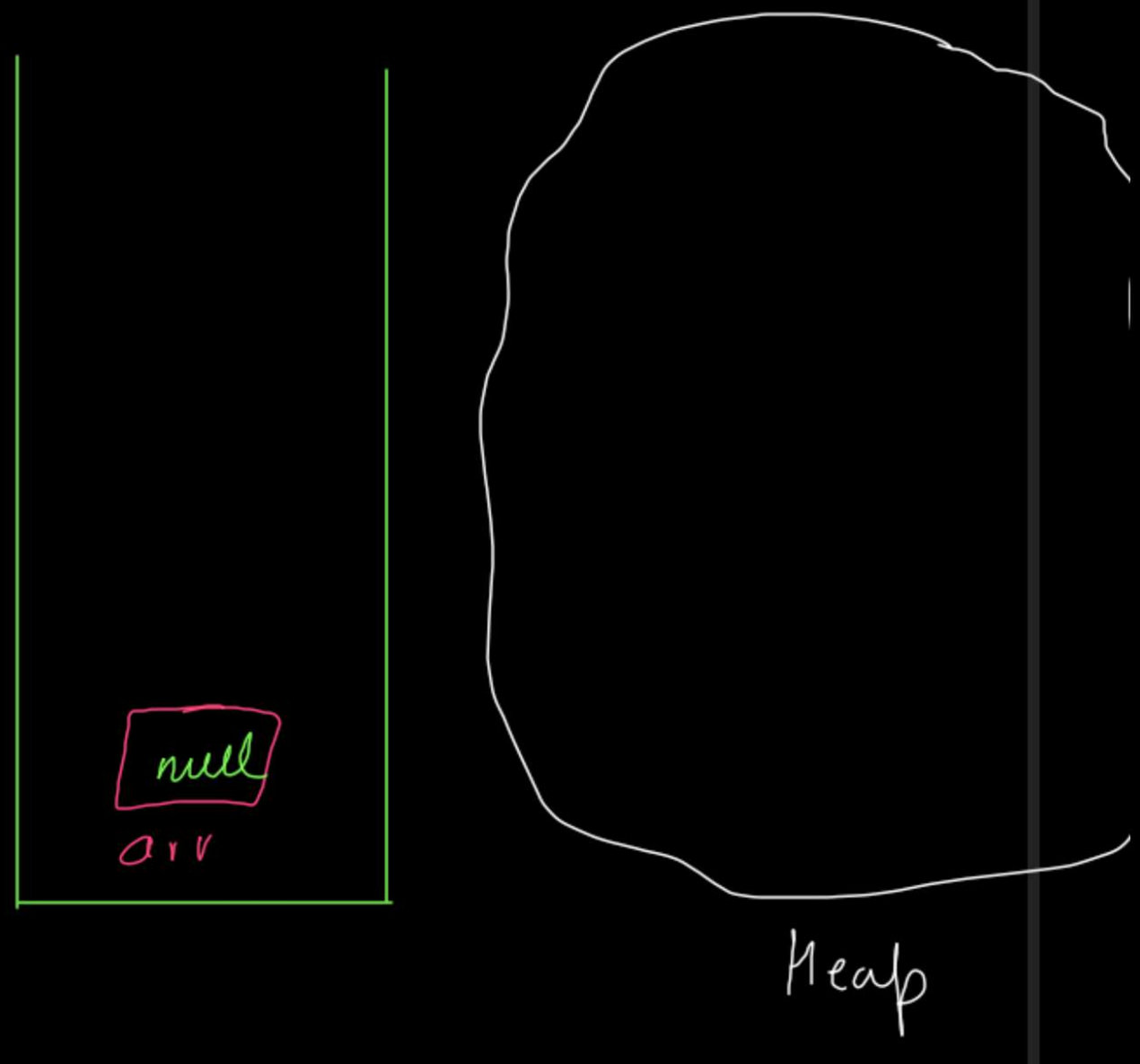
Inbuilt exception handling

```
int [] arr;   // array declaration
arr = new int[10];   // array intialization
```

Array declaration (Memory Mapping)

```java
import java.util.*;
public class Main {
    public static void main(String[] args) {
        int[] arr; //declaration

    }
}
```

null → it is a type of value
that is the initial value
for all the
arrays / objects

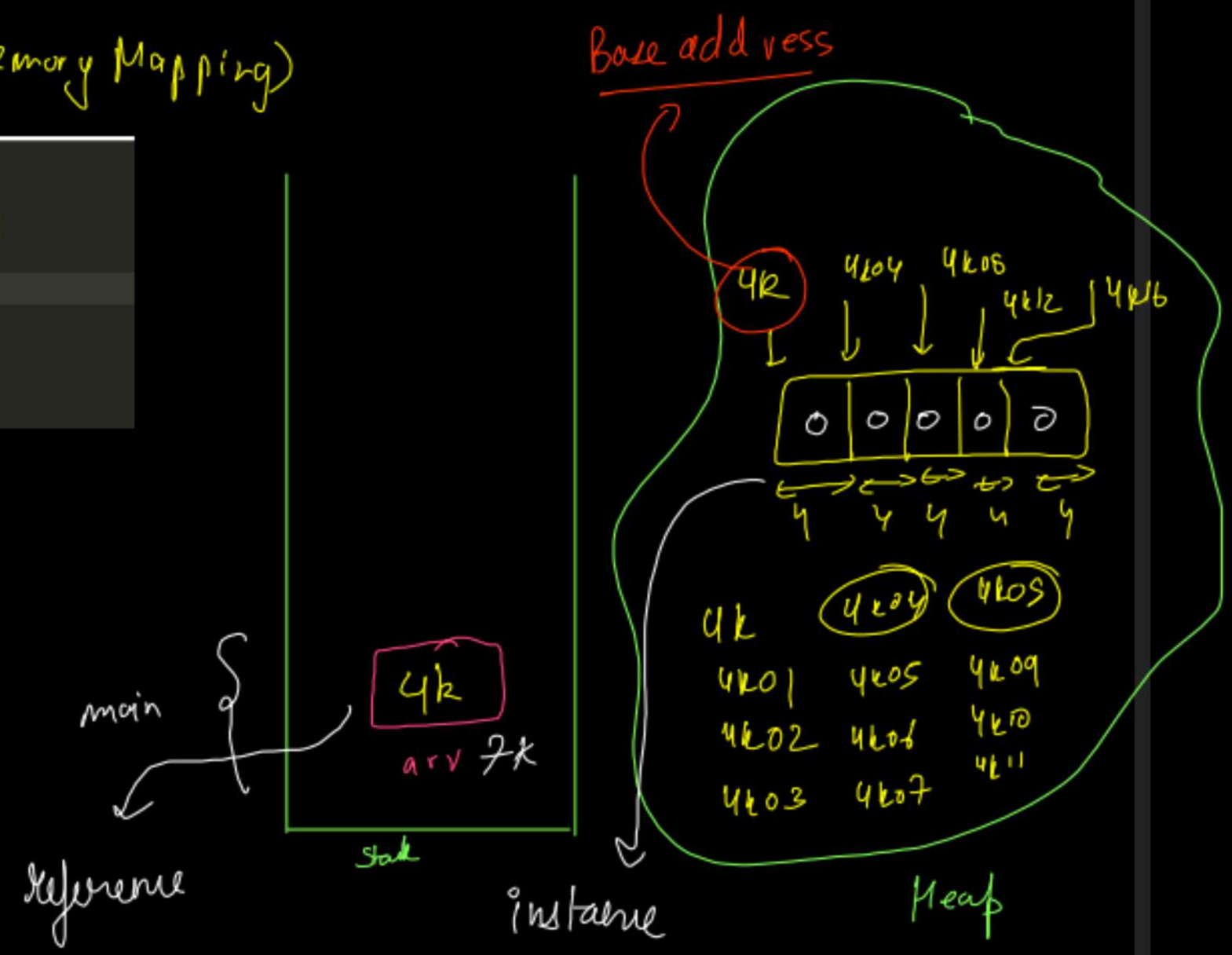


Array Initialization (Memory Mapping)

```java
import java.util.*;
public class Main {
    public static void main(String[] args) {
        int[] arr; //declaration
        arr = new int[5]; //initialization
    }
}
```

Allocate memory in the heap

1 2 3 4 5



The variable that stores the address of actual memory & itself exists in the Stack is called a reference.

The actual memory (formed in heap) is called instance.

Size of a reference variable can never be known in Java.
(In C++ it is = sizeof(int) = 4bytes)

** # Array will always be created inside heap in Java.

When we don't initialize an array, the reference will store null by default. Null means that the reference is currently empty & does not store any address.

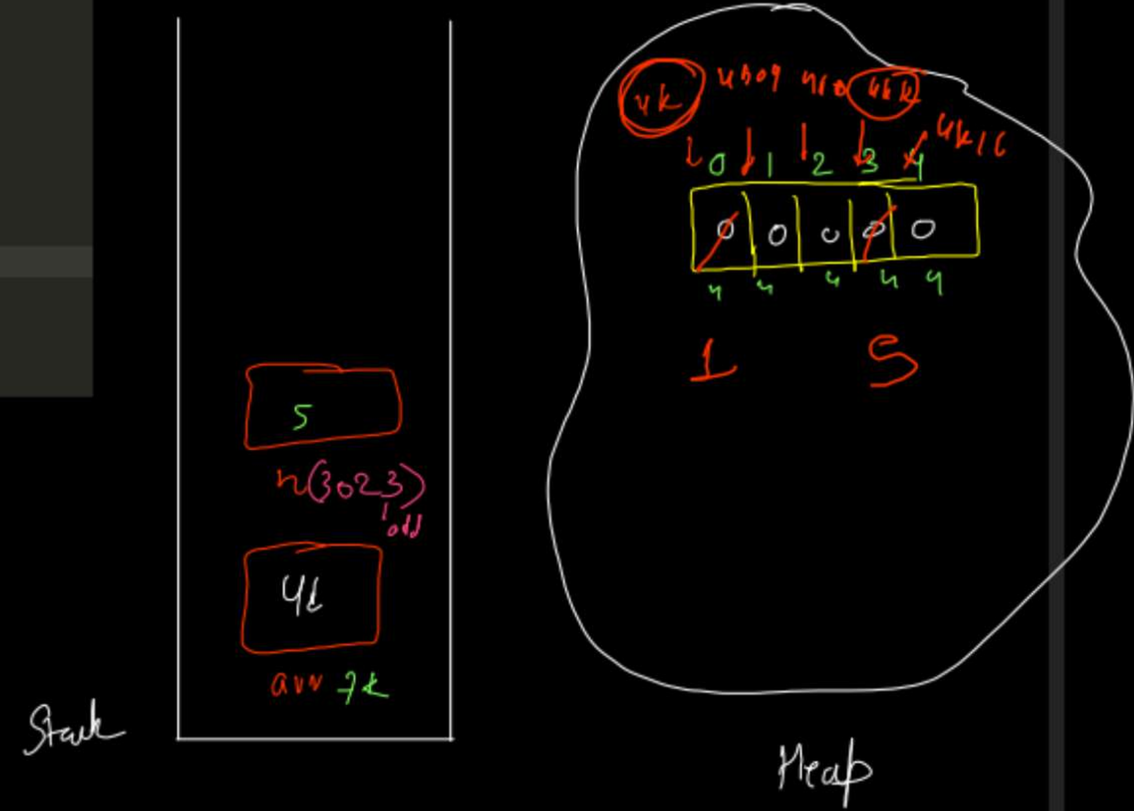
When we initialize an array but have not stored any values in it, Java stores 0 by default. (C++ stores garbage value)

In case of Boolean array, false is stored by default.

```java
import java.util.*;
public class Main {
    public static void main(String[] args) {
        int[] arr; //declaration
        int n = 5;
        arr = new int[n]; //initialization

        arr[0] = 1;
        arr[3] = 5;
    }
}
```

Min index = 0

Max index = length of Array - 1



LOC = Base address + Size of DT * Index → O(1)

4k + 4 * 3 = 4k + 12 = 4k2

Traversing the Array

```java
import java.util.*;
public class Main {
    public static void main(String[] args) {
        int[] arr; //declaration
        int n = 5;
        arr = new int[n]; //initialization

        for(int i=0;i<n;i++) {
            arr[i] = i+1;
        }

        for(int i=0;i<n;i++) {
            System.out.print(arr[i] + " ");
        }
    }
```

```java
import java.util.*;
public class Main {
    public static void main(String[] args) {
        int[] arr; //declaration
        int n = 5;
        arr = new int[n]; //initialization

        for(int i=0;i<n;i++) {
            arr[i] = i+1;
        }

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

```
Finished in 80 ms
[I@2503dbd3
```

This is a hashcode

Whenever you try to print a reference, toString() function is automatically called

```java
import java.util.*;
public class Main {
    public static void main(String[] args) {
        int[] arr; //declaration
        int n = 5;
        arr = new int[n]; //initialization

        for(int i=0;i<n;i++) {
            arr[i] = i+1;
        }

        System.out.println(arr.toString());
    }
}
```

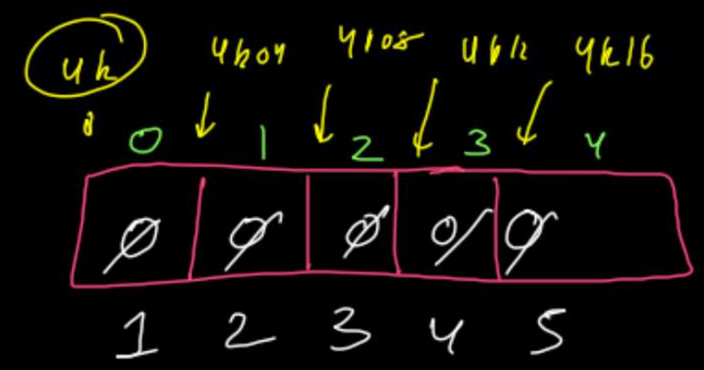
toString is automatically invoked.

Getting Started (Lecture-10)

- → Array Length Property
- → Array as a function parameter & as a return type. → Shallow & deep Copy concepts
- → Linear Search
- → Find Min & Max
- → Reverse an Array
- → Left Rotate
- → Right Rotate
- → Bar Chart
- → Binary Search

Length Property

```java
import java.util.*;
public class Main {
    public static void main(String[] args) {
        int[] arr; //declaration
        int n = 5;
        arr = new int[n]; //initialization

        for(int i=0;i<arr.length;i++) {
            arr[i] = i+1;
        }

        for(int i=0;i<arr.length;i++) {
            System.out.print(arr[i] + " ");
        }
    }
}
```



arr.length = 5

LOC = Base add + DT * index

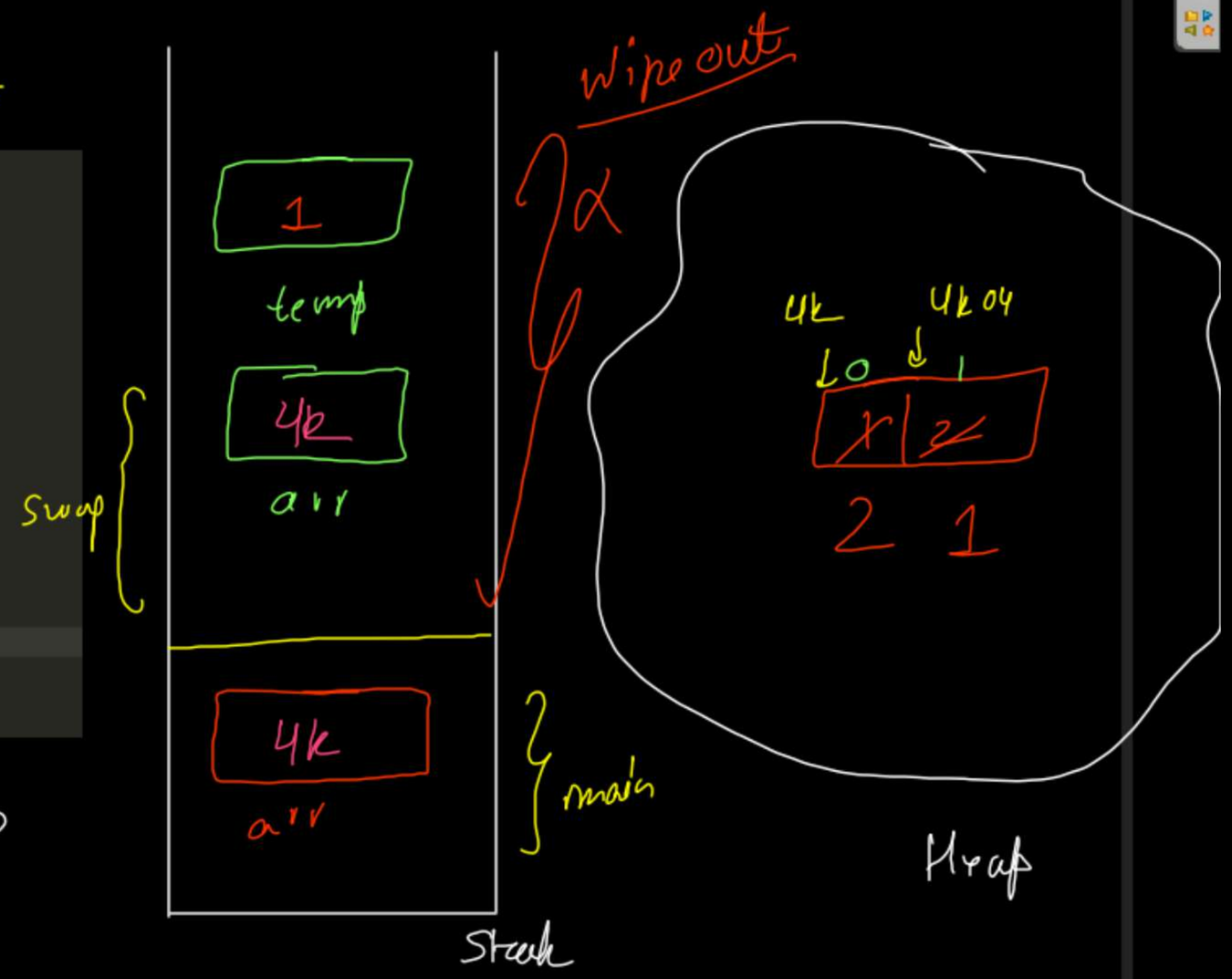
4k + 4*3 = 4k+12
4k15

Array as a function Parameter

```java
import java.util.*;
public class Main {
    public static void main(String[] args) {
        int[] arr = {1,2}; //another method for array creation

        System.out.println("arr[0] = " + arr[0] + " arr[1]" + arr[1]);

        //function call
        swap(arr);

        System.out.println("arr[0] = " + arr[0] + " arr[1]" + arr[1]);
    }

    public static void swap(int[] arr) {
        int temp = arr[0];
        arr[0] = arr[1];
        arr[1] = temp;
    }
}
```

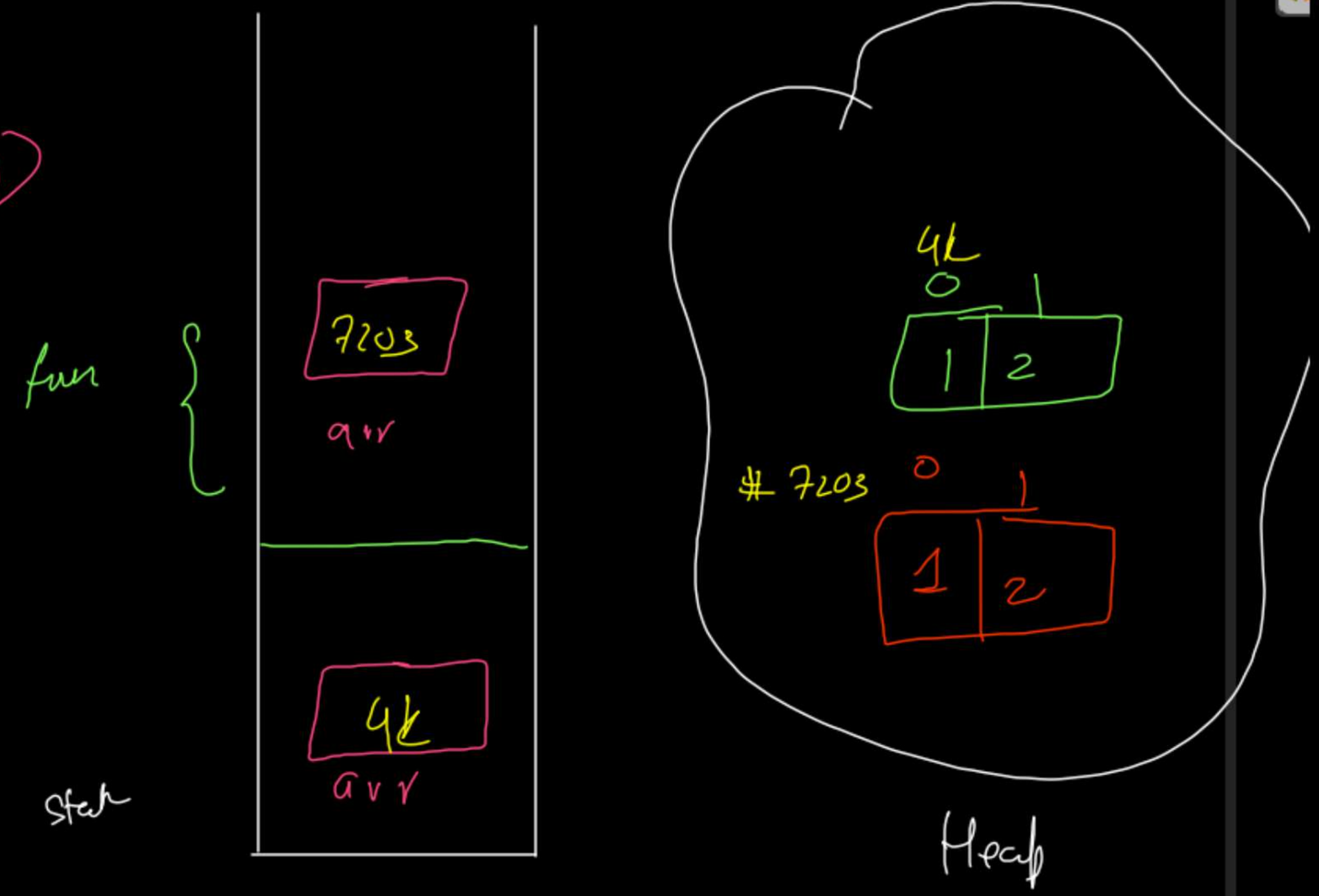
1 2

This is shallow copy.



Deep Copy

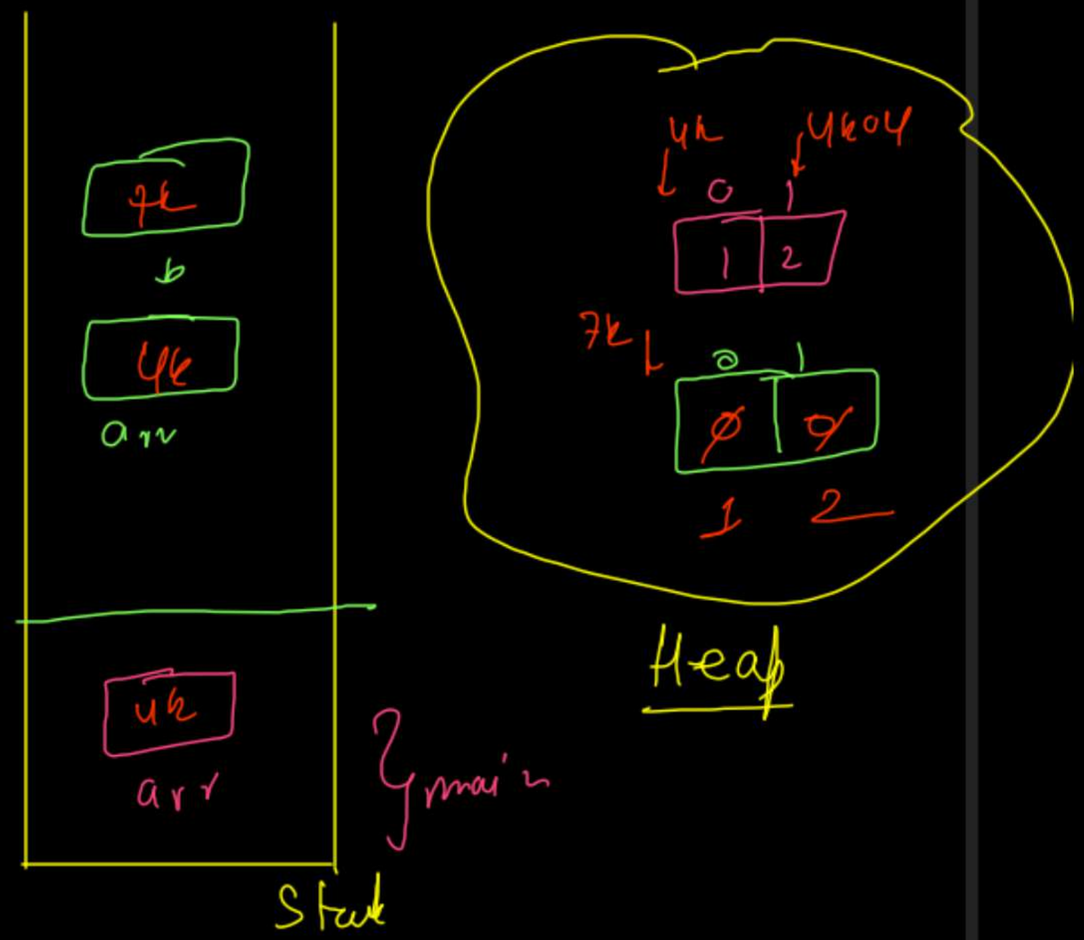
(If it were to happen)



How to Deep Copy an Array

```java
import java.util.*;
public class Main {
    public static void main(String[] args) {
        int[] arr = {1,2}; //another method for array creation

        System.out.println("arr[0] = " + arr[0] + " arr[1]" + arr[1]);

        //function call
        swap(arr);

        System.out.println("arr[0] = " + arr[0] + " arr[1]" + arr[1]);
    }

    public static void swap(int[] arr) {

    }
}
```

```java
    public static void swap(int[] arr) {
        int[] b = new int[arr.length];

        for(int i=0;i<arr.length;i++) {
            b[i] = arr[i];
        }
    }
```



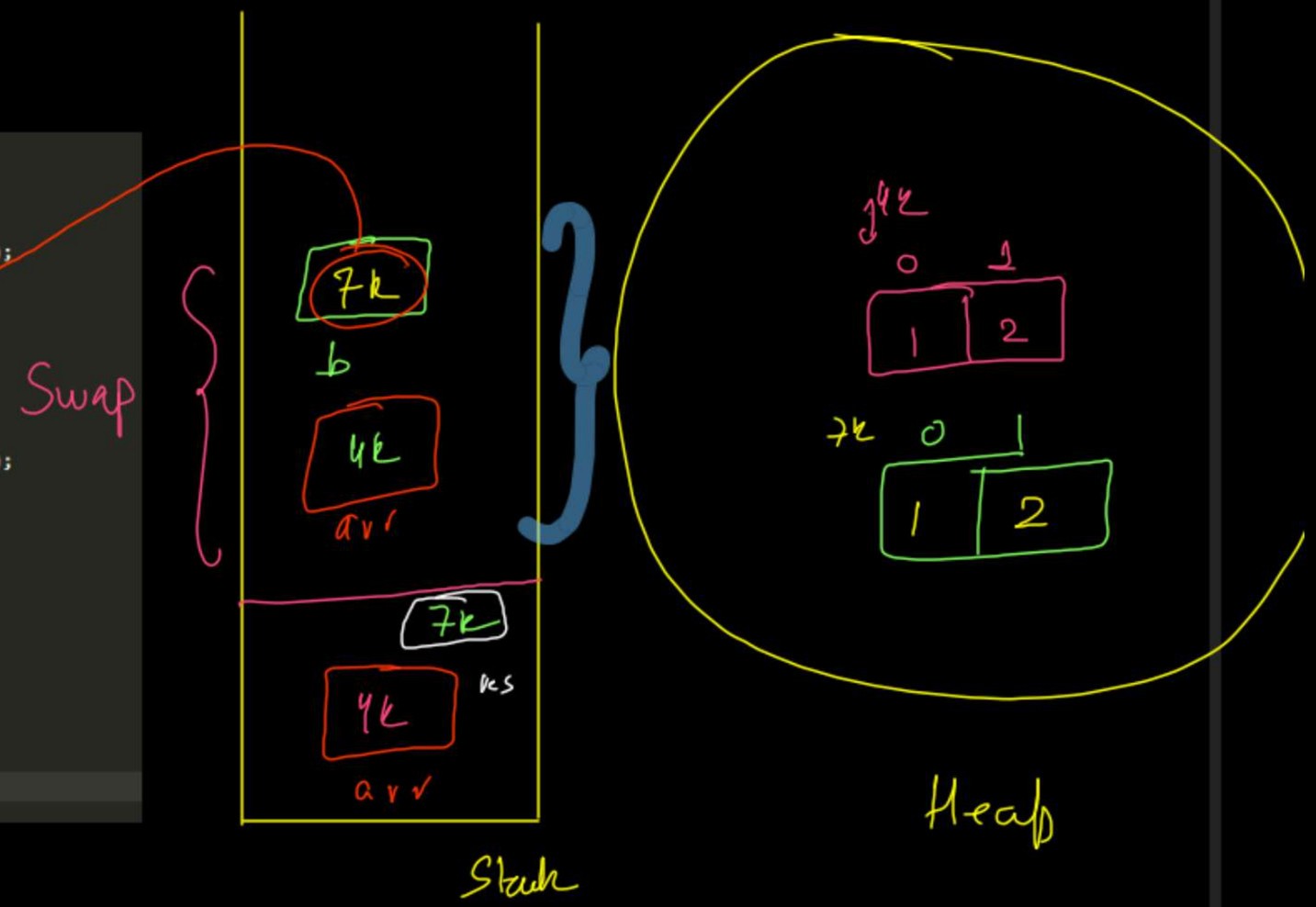

```
int [] a = new int [10];      → 4k
int [] b = new int [10000];   → 7k

fun(a);   (4k)        fun(b);   (7k)
```

Since only address is passed, both will take same time.

Array as a return type -

```java
public class Main {
    public static void main(String[] args) {
        int[] arr = {1,2}; //another method for array creation

        System.out.println("arr[0] = " + arr[0] + " arr[1] = " + arr[1]);

        //function call
        swap(arr);

        int[] res = swap(arr);

        System.out.println("res[0] = " + res[0] + " res[1] = " + res[1]);
    }

    public static int[] swap(int[] arr) {
        int[] b = new int[arr.length];

        for(int i=0;i<arr.length;i++) {
            b[i] = arr[i];
        }

        return b;
    }
}
```

```
Finished in 77 ms
arr[0] = 1 arr[1] = 2
res[0] = 1 res[1] = 2
```



Search an Element in an array

Basic Accuracy: **50.0%** Submissions: **100k+** Points: **1**

Given an integer array and another integer element. The task is to find if the given element is present in array or not.

Linear Search $O(N)$

↳ Unsorted

$i \longrightarrow$ return

0	1	2	3	4	5	6
9	8	6	3	4	$\boxed{7}$	2

$X = 7$

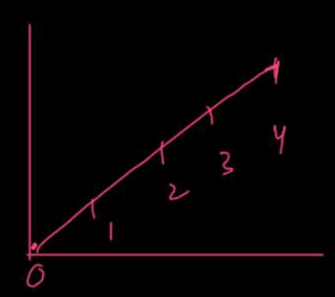
Successful Search

i = arr-length

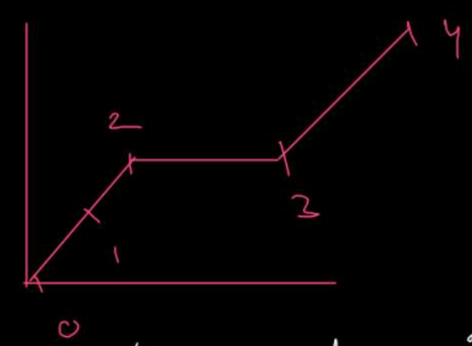
0	1	2	3	4	5	6
9	8	6	3	4	7	2

$X = 5$

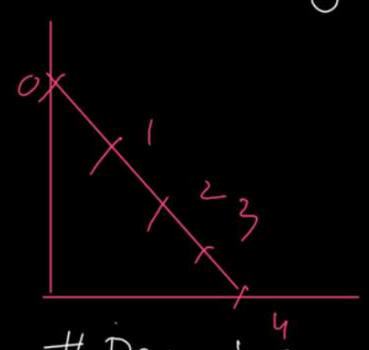
Unsuccessful Search



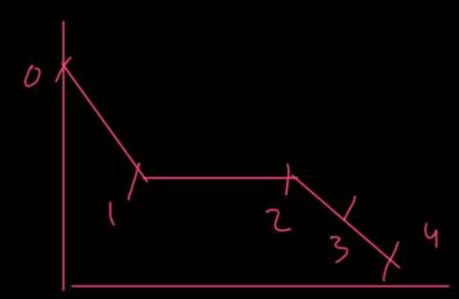
Increasing



Non-decreasing



Decreasing



Non increasing

```java
class Solution{

    static int search(int arr[], int N, int X)
    {

        // Your code here

        for(int i=0;i<N;i++) {
            if(arr[i] == X) return i;
        }

        return -1;
    }

}
```

Linear Search

→ It is usually used when the array (data) is unsorted.

→ Time Complexity

Best Case : O(1)

Worst Case : O(N)

Avg Case : O(N)

Find minimum and maximum element in an array

Basic Accuracy: **59.4%** Submissions: **100k+** Points: **1**

Given an array **A** of size **N** of integers. Your task is to find the **minimum and maximum** elements in the array.

$$\left.\begin{aligned} min &= \infty \\ max &= -\infty \end{aligned}\right\} \text{Identities}$$

$$\begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ \{8, & 6, & 3, & 9, & 4, & 2, & 7\} \end{array}$$

0 → additive identity.

1 → multiplicative identity

Identity is a term using which when we perform an operation with N (any no), we get N itself.

$N + 0 = N$

$N * 1 = N$

Minimum Identity

$\infty \rightarrow$ is the min identity

$100, \infty \rightarrow 100$

$100000, \infty \rightarrow 100000$

Maximum Identity

$-\infty \rightarrow$ is the max identity

$-10000000, -\infty \rightarrow$ Max

i i i i i i i i

0 1 2 3 4 5 6

{8, 6, 3, 9, 4, 2, 7}

max = ~~-∞~~ ~~8~~ 9

min = ~~∞~~ ~~8~~ ~~6~~ ~~3~~ 2

Max & min store the max & min values so far.

```
class Compute
{
    static pair getMinMax(long a[], long n)
    {
        //Write your code here
        long max = Long.MIN_VALUE;
        long min = Long.MAX_VALUE;

        for(int i=0;i<a.length;i++) {
            if(a[i] > max) max = a[i];
            if(a[i] < min) min = a[i];
        }

        return new pair(min,max);
    }
}
```

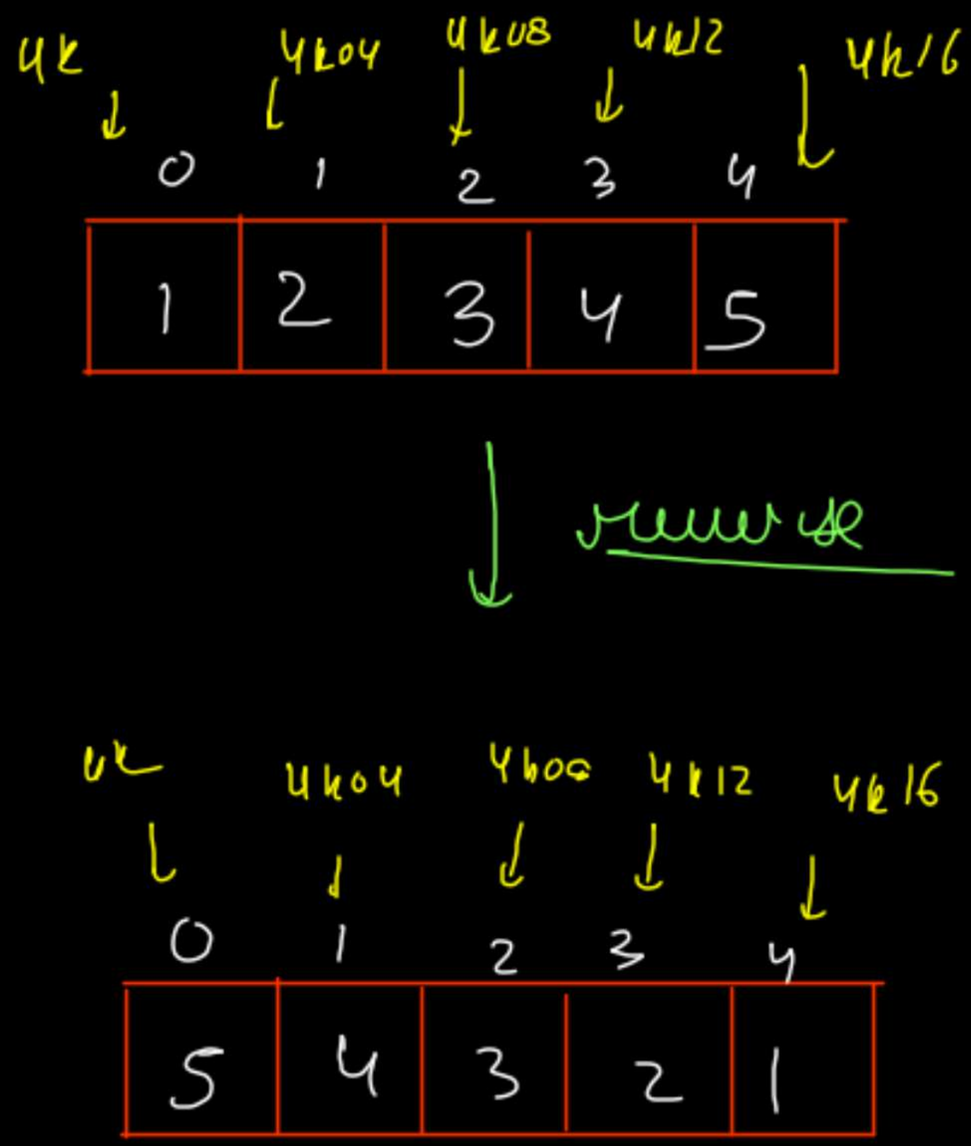
Reverse an Array

School Accuracy: **40.0%** Submissions: **100k+** Points: **0**

Given an array **A** of size **N**, print the reverse of it.

Example:

Input:
1
4
1 2 3 4
Output:
4 3 2 1


4k
4k04
4k08
4k12
4k16
0
1
2
3
4
1
2
3
4
5
reverse
4k
4k04
4k08
4k12
4k16
0
1
2
3
4
5
4
3
2
1

lo hi

0	1	2	3	4
~~1~~	~~2~~	3	~~4~~	~~5~~
5	4		2	1

```
int temp = a[lo];
a[lo] = a[hi];
a[hi] = temp;
```

hi lo

0	1	2	3	4	5
~~1~~	~~2~~	~~3~~	~~4~~	~~5~~	~~6~~
6	5	4	3	2	1

2 (T)
5 (N)
1 2 3 4 5
6
1 2 3 4 5 6

1	2	3	4	5

5 4 3 2 1

1	2	3	4	5	6

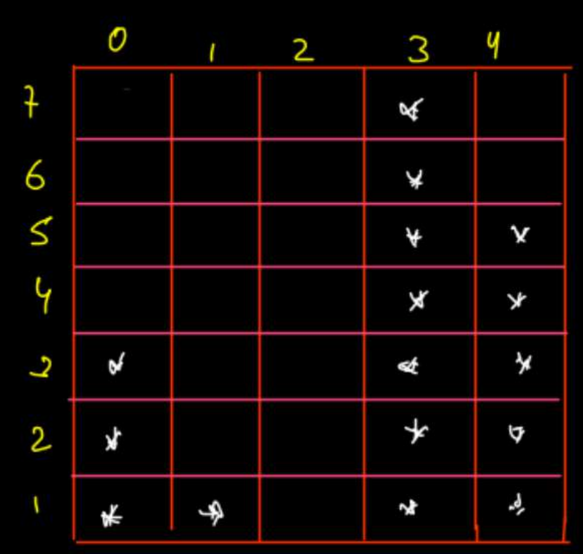
6 5 4 3 2 1

```java
public static void reverse(int[] arr) {
    int lo = 0;
    int hi = arr.length -1;

    while(lo < hi) {
        int temp = arr[lo];
        arr[lo] = arr[hi];
        arr[hi] = temp;
        lo++;
        hi--;
    }
}

public static void display(int[] arr) {
    for(int i=0;i<arr.length;i++) {
        System.out.print(arr[i] + " ");
    }
    System.out.println();
}
```

```java
public static void main (String[] args) {
    //code

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

    while(t-- > 0) {
        int n = scn.nextInt();
        int[] arr = new int[n];

        for(int i=0;i<n;i++) {
            arr[i] = scn.nextInt();
        }

        reverse(arr);
        display(arr);
    }
}
}
```

Barchart

0	1	2	3	4
3	1	0	7	5




```java
for (i=7; i>=1; i--)
{
    for(j=0; j<4; j++) {

    }
```

```java
public static void main(String[] args) throws Exception {
    // write your code here
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();

    int[] arr = new int[n];

    for(int i=0;i<n;i++) {
        arr[i] = scn.nextInt();
    }

    int max = Integer.MIN_VALUE;

    for(int i=0;i<n;i++) {
        max = (int)Math.max(max,arr[i]);
    }

    for(int i=max;i>=1;i--) {
        for(int j=0;j<n;j++) {
            if(arr[j] >= i) {
                System.out.print("*\t");
            } else {
                System.out.print("\t");
            }
        }
        System.out.println();
    }

 }

}
```

Lecture - 11 (Arrays Continued)

1. Left Rotate
2. Right Rotate

} Reversal Algorithm

3. Binary Search (A glimpse into BS topic)
4. Add 2 Arrays
5. Add 1 to an Array (H.W)
6. Subtract 2 Arrays
7. Subarrays of an Array

Left Rotate

Rotate Array

Basic Accuracy: **59.87%** Submissions: **100k+** Points: **1**

Given an unsorted array **arr[]** of size **N**, rotate it by **D** elements (Anti - clockwise).

Input:
The first line of the input contains **T** denoting the number of testcases. First line of each test case contains two space separated elements, **N** denoting the size of the array and an integer **D** denoting the number size of the rotation. Subsequent line will be the N space separated array elements.

$\{1, 2, 3, 4, 5\}$

$k = 2$

$\{3, 4, 5, 1, 2\}$

$$\{\overset{0}{1}, \overset{1}{2}, \overset{2}{3}, \overset{3}{4}, \overset{4}{5}\}$$

$N = 5$

$k = 0 \quad \{1, 2, 3, 4, 5\} \quad k = 5$

$k = 1 \quad \{2, 3, 4, 5, 1\} \quad k = 6$

$k = 2 \quad \{3, 4, 5, 1, 2\} \quad k = 7$

$k = 3 \quad \{4, 5, 1, 2, 3\} \quad k = 8$

$k = 4 \quad \{5, 1, 2, 3, 4\} \quad k = 9$

$\# \; k = k \% N$

$\# \; \underbrace{k = k + N}_{\text{if } k < 0}$

$$\{\overset{0}{1}, \overset{1}{2}, \overset{2}{3}, \overset{3}{4}, \overset{4}{5}\}$$

$k = 2$

① Reverse(arr, $\overset{lo}{0}$, $\overset{hi}{k-1}$) $\quad \{\overset{0}{2}, \overset{1}{1}, \overset{2}{3}, \overset{3}{4}, \overset{4}{5}\}$

② Reverse(arr, $\overset{lo}{k}$, $\overset{hi}{N-1}$) $\quad \{\overset{0}{2}, \overset{1}{1}, \overset{2}{5}, \overset{3}{4}, \overset{4}{3}\}$

③ Reverse(arr, $\overset{lo}{0}$, $\overset{hi}{N-1}$) $\quad \{\overset{0}{3}, \overset{1}{4}, \overset{2}{5}, \overset{3}{1}, \overset{4}{2}\}$

```java
public static void reverse(int[] arr,int lo, int hi) {
    while(lo < hi) {
        int temp = arr[lo];
        arr[lo] = arr[hi];
        arr[hi] = temp;
        lo++;
        hi--;
    }
}

public static void rotate(int[] arr, int k) {

    int n = arr.length;

    reverse(arr,0,k-1);
    reverse(arr,k,n-1);
    reverse(arr,0,n-1);
}

public static void display(int[] arr) {
    for(int i=0;i<arr.length;i++) {
        System.out.print(arr[i] + " ");
    }
    System.out.println();
}
```

```java
    public static void main (String[] args) {
        //code

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

        while(t-- > 0) {
            int n = scn.nextInt();
            int k = scn.nextInt();

            int[] arr = new int[n];

            for(int i=0;i<n;i++) {
                arr[i] = scn.nextInt();
            }

            rotate(arr,k);
            display(arr);
        }
    }
}
```

$O(N)$

189. Rotate Array

Medium 11203 1397 Add to List Share

Given an array, rotate the array to the right by `k` steps, where `k` is non-negative.

Example 1:

```
Input: nums = [1,2,3,4,5,6,7], k = 3
Output: [5,6,7,1,2,3,4]
Explanation:
rotate 1 steps to the right: [7,1,2,3,4,5,6]
rotate 2 steps to the right: [6,7,1,2,3,4,5]
rotate 3 steps to the right: [5,6,7,1,2,3,4]
```

N-k N-1

0 1 2 3 4 5 6

{1, 2, 3, 4, 5, 6, 7} k=3

0 1 2 3 4 5 6

{5, 6, 7, 1, 2, 3, 4}

0 N-k-1

Reverse(arr, N-k, N-1) → {1, 2, 3, 4, 7, 6, 5}

Reverse(arr, 0, N-k-1) → {4, 3, 2, 1, 7, 6, 5}

Reverse(arr, 0, N-1) → {5, 6, 7, 1, 2, 3, 4}

N-k-1 N-k N-1

0 1 2 3 4 5 6

{1, 2, 3, 4, 5, 6, 7} k=3

{5, 6, 7, 1, 2, 3, 4}

```java
class Solution {

     public void reverse(int[] arr,int lo, int hi) {
        while(lo < hi) {
            int temp = arr[lo];
            arr[lo] = arr[hi];
            arr[hi] = temp;
            lo++;
            hi--;
        }
    }

    public void rotate(int[] nums, int k) {

        int n = nums.length;

        k = k%n;

        reverse(nums,n-k,n-1);
        reverse(nums,0,n-k-1);
        reverse(nums,0,n-1);
    }
}
```

Binary Search

704. Binary Search

Easy 6477 138 Add to List Share

Given an array of integers `nums` which is sorted in ascending order, and an integer `target`, write a function to search `target` in `nums`. If `target` exists, then return its index. Otherwise, return `-1`.

You must write an algorithm with `O(log n)` runtime complexity.

while (lo < hi)

target = 40

~~lo~~					mid	lo				hi
0	1	2	3	4	5	6	7	8	9	10
-40	-30	-20	-10	10	20	30	40	60	70	80

$$mid = \frac{(lo+hi)}{2} = \frac{0+10}{2} = 5$$

```
if (arr[mid] < target)
    lo = mid + 1;
```

						lo	hi	mid		
0	1	2	3	4	5	6	7	8	9	10
-40	-30	-20	-10	10	20	30	40	60	70	80

$$mid = \frac{lo+hi}{2} = \frac{16}{2} = 8$$

```
if (arr[mid] > target)
    hi = mid - 1;
```

mid lo

~~lo~~ hi

0	1	2	3	4	5	6	7	8	9	10
-40	-30	-20	-10	10	20	30	40	60	70	80

$$mid = \frac{lo + hi}{2} = \frac{6+7}{2} = \frac{13}{2} = 6$$

$$lo = mid + 1$$

lo mid

hi

0	1	2	3	4	5	6	7	8	9	10
-40	-30	-20	-10	10	20	30	40	60	70	80

$$mid = \frac{lo + hi}{2} = \frac{7+7}{2} = 7$$

$$\approx \log_2 N$$

```
if (arr[mid] == target)
    return mid;
```

Unsuccessful Search

target = 50

hi lo

0	1	2	3	4	5	6	7	8	9	10
-40	-30	-20	-10	10	20	30	40	60	70	80

mid = $\frac{0+10}{2}$ = 5

while(lo ≤ hi)

```
class Solution {
    public int search(int[] nums, int target) {
        int lo = 0;
        int hi = nums.length -1;

        while(lo <= hi) {
            int mid = (lo + hi)/2;

            if(nums[mid] == target) return mid;
            else if(nums[mid] < target) lo = mid + 1;
            else hi = mid -1;
        }

        return -1; //only hits when unsucessful search
    }
}
```

Overflow Condition

$$mid = (lo + hi) / 2$$

∞ ∞

Out of the Range

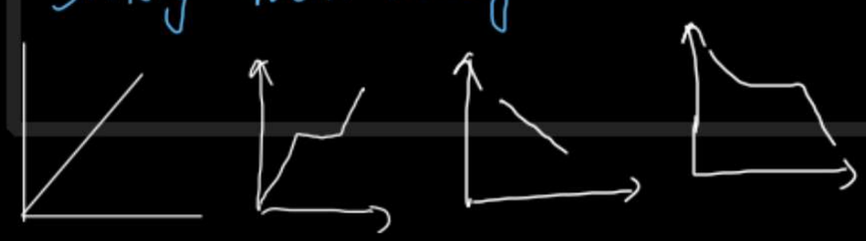
```
class Solution {
    public int search(int[] nums, int target) {
        int lo = 0;
        int hi = nums.length -1;

        while(lo <= hi) {
            int mid = lo + (hi-lo)/2;

            if(nums[mid] == target) return mid;
            else if(nums[mid] < target) lo = mid + 1;
            else hi = mid -1;
        }

        return -1; //only hits when unsucessful search
    }
}
```

Binary Search

Monotonic Array

Increasing, Decreasing, Binary

Strictly, Non, Strictly, Non



0000 ... 111 ..

111 ... 000 ...

Note: This is not complete.

We will complete it when we will study BS.

Sum of two numbers represented as arrays

Easy Accuracy: **43.7%** Submissions: **5244** Points: **2**

Given two numbers represented by two different arrays **A** and **B**. The task is to find the sum array. The sum array is an array representation of addition of two input arrays.

Example 1:

Input:
N = 3, M = 3
A[] = {5, 6, 3}
B[] = {8, 4, 2}
Output: 1 4 0 5
Explanation: As 563 + 842 = 1405.

i

0 { 9, 9, 3 } → a

0 1 2

j

0 { 1, 5 } → b

0 1

k

0 1 2 3

{ 1 0 0 8 } → c

ca = ~~1~~ 1

```
if (a[i] + b[j] + ca < 10) {
    c[k] = a[i] + b[j] + ca;
    ca = 0
}
```

```
if (a[i] + b[j] + ca >= 10) {
    c[k] = (a[i] + b[j] + ca) % 10;
    ca = (a[i] + b[j] + ca) / 10
}
```

Array can contain one 0 in the beginning.

{9, 3, 0}

{2, 3}

{0, 9, 5, 3}

0 1 2 3

{9, 5, 3}

{9, 5, 3}

```java
int[] sum(int[] a, int[] b) {
    int size = (a.length > b.length) ? a.length + 1 : b.length +
    int[] res = new int[size];

    int i = a.length -1, j = b.length-1, k = size-1;
    int ca = 0;

    while(i >= 0 || j >= 0 || ca > 0) {
        int d1 = (i >= 0) ? a[i] : 0;
        int d2 = (j >= 0) ? b[j] : 0;

        if(d1 + d2 + ca < 10) {
            res[k] = d1 + d2 + ca;
            ca = 0;
        } else {
            res[k] = (d1 + d2 + ca) % 10;
            ca = (d1 + d2 + ca) / 10;
        }

        i--;j--;k--;
    }

    return res;

}

ArrayList<Integer> findSum(int a[], int b[]) {
    // code here
    int[] res = sum(a,b);

    ArrayList<Integer> list = new ArrayList<>();

    for(int i=0;i<res.length;i++) {
        if(i == 0 && res[i] == 0) {
            continue;
        } else {
            list.add(res[i]);
        }
    }

    return list;
```

You are screen sha

Lecture - 12

Subtraction of 2 Arrays

Subarrays of an Array

2-D Arrays (Matrices)

Demo and Memory Mapping Concepts.

#Subtraction of 2 Arrays

Difference Of Two Arrays

Easy

Prev Next

1. You are given a number n1, representing the size of array a1.
2. You are given n1 numbers, representing elements of array a1.
3. You are given a number n2, representing the size of array a2.
4. You are given n2 numbers, representing elements of array a2.
5. The two arrays represent digits of two numbers.
6. You are required to find the difference of two numbers represented by two arrays and print the arrays. a2 - a1

Assumption - number represented by a2 is greater.

j ~~j~~ ~~j~~ ~~j~~ ~~j~~
0 1 2 3

$\{1, 0, 2, 3\} \rightarrow a_2$

i ~~i~~ ~~i~~ ~~i~~
0 1 2

$\{4, 5, 1\} \rightarrow a_1$

$b_0 = \cancel{0}\ -1$

k ~~k~~ ~~k~~ ~~k~~ ~~k~~
0 1 2 3

$\{0\ 5\ 7\ 2\}$

```
if (d2 - d1 + b0 < 0){
    d = d2 - d1 + b0 + 10;
    b0 = -1;
} else {
    d = d2 - d1 + b0;
    b0 = 0;
}
```

$\{1, 0, 2, 3\}$

$\{0\}$

This is the condition when we can have the max size of res array. It is = size of larger array

$$\{\overset{0}{1},\overset{1}{0},\overset{2}{0},\overset{3}{0},\overset{4}{0}\}$$

$$\{\overset{0}{9},\overset{1}{0},\overset{2}{0}\}$$

$$\{0\ 9\ 1\ 0\ 0\}$$

$$\{1,0,0,0\}$$

$$\{9,9,9\}$$

$$\{\overset{\cancel{idx}}{1},\ \overset{idx}{0},\ \overset{\cancel{idx}}{0},\ \overset{idx}{2}\}$$

There can be multiple 0s at the start of res.

If both Nos are equal

{9, 9, 8}

{9, 9, 8}

~~idx~~ ~~idx~~ ~~idx~~ idx → index out of bound

{0, 0, 0}

```java
public static int[] diff(int[] arr1,int[] arr2) {
    int[] res = new int[arr2.length]; //size of the larger array
    int i = arr1.length-1, j = arr2.length -1, k = res.length-1;

    int bo = 0;

    while(j >= 0) {
        int d1 = (i >= 0) ? arr1[i] : 0; //this is to be safe from index out of bounds
        int d2 = arr2[j];

        if(d2 - d1 + bo < 0) {
            res[k] = d2 - d1 + bo + 10;
            bo = -1;
        } else {
            res[k] = d2 - d1 + bo;
            bo = 0;
        }

        i--;j--;k--;
    }

    return res;
```

```java
    //differnce
    int[] res = diff(arr1,arr2);

    int idx = 0;

    while(idx < res.length && res[idx] == 0) {
        idx++;
    }

    if(idx == res.length) {
        System.out.println("0");
    }

    while(idx < res.length) {
        System.out.println(res[idx]);
        idx++;
    }
```

Subarrays of an Array

Subarray Problem

Easy

Prev Next

1. You are given an array of size 'n' and n elements of the same array.
2. You are required to find and print all the subarrays of the given array.
3. Each subarray should be space seperated and on a seperate lines.

Refer to sample input and output.

Sample Input

```
3
10
20
30
```

Sample Output

```
10
10 20
10 20 30
20
20 30
30
```

$\{\overset{0}{10}, \overset{1}{20}, \overset{2}{30}\}$

$\{\overset{0}{10}, \overset{1}{30}\}$ ✗ this is not a subarray.

Subarray is a part of an array that contains elements in continuation.

$\{\overset{0}{10}, \overset{1}{20}, \overset{2}{30}\}$

1 $\{10\}$

2 $\{10, 20\}$

3 $\{10, 20, 30\}$

} Starting from index 0

4 $\{20\}$

5 $\{20, 30\}$

} Starting from index 1

6 $\{30\}$

} Starting from index 2

Number of subarrays of an Array

$$= \frac{N(N+1)}{2}$$

Here N is the size of the array

n, sp, ep

0 1 2

{10, 20, 30}

1 {10}

2 {10, 20}

3 {10, 20, 30}

} Starting from index 0

4 {20}

5 {20, 30}

} Starting from index 1

6 {30}

} Starting from index 2

```
for (int sp = 0; sp < n; sp++)
{
  for (int ep = i; ep < n; ep++)
  {
    for (int k = sp; k ≤ ep; k++)
    {
      Print( arr[k] + " ")
    }
  }
}
```

→ Leave a lin

i

1 sp

{ 10, 20, 30, 40, 50, 60, 70 }

30

30, 40

30, 40, 50

30, 40, 50, 60

30, 40, 50, 60, 70

```java
for(int sp=0;sp<arr.length;sp++) {
    for(int ep=sp;ep<arr.length;ep++) {
        for(int k=sp;k<=ep;k++) {
            System.out.print(arr[k] + "\t");
        }
        System.out.println();
    }
}
```

All subsets

All subarrays

All paths

Brute force

2-D Arrays

	1	2	3	4
1	a_{11}	a_{12}	a_{13}	a_{14}
2	a_{21}	a_{22}	a_{23}	a_{24}
3	a_{31}	a_{32}	a_{33}	a_{34}

↓

Matrix (Maths)

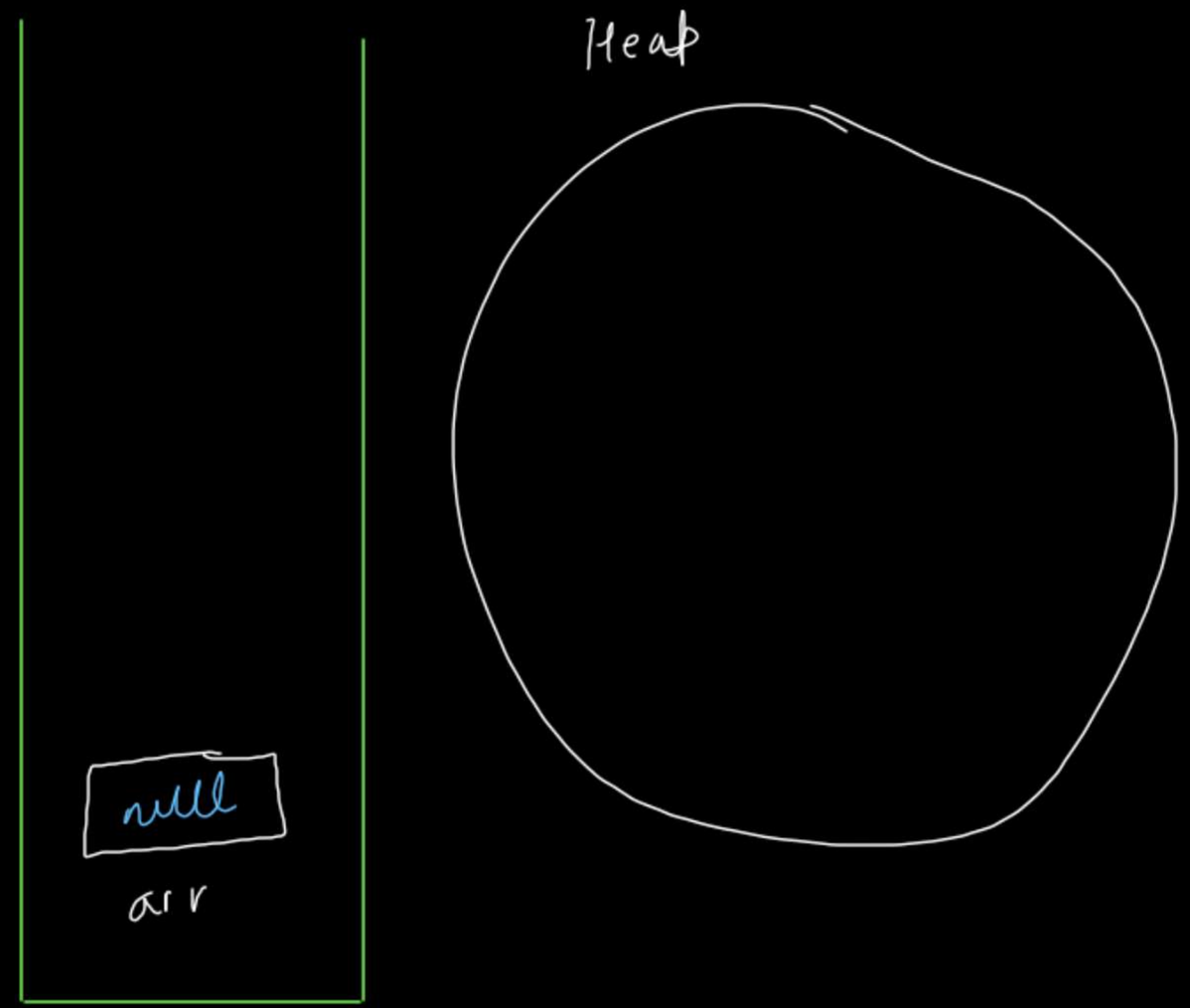
	0	1	2	3
0	a_{00}	a_{01}	a_{02}	a_{03}
1	a_{10}	a_{11}	a_{12}	a_{13}
2	a_{20}	a_{21}	a_{22}	a_{23}

↓

2-D Array

Declare

```
int[][] arr;
```



Initialization

```
int [][] arr;
arr = new int [3][4];
```

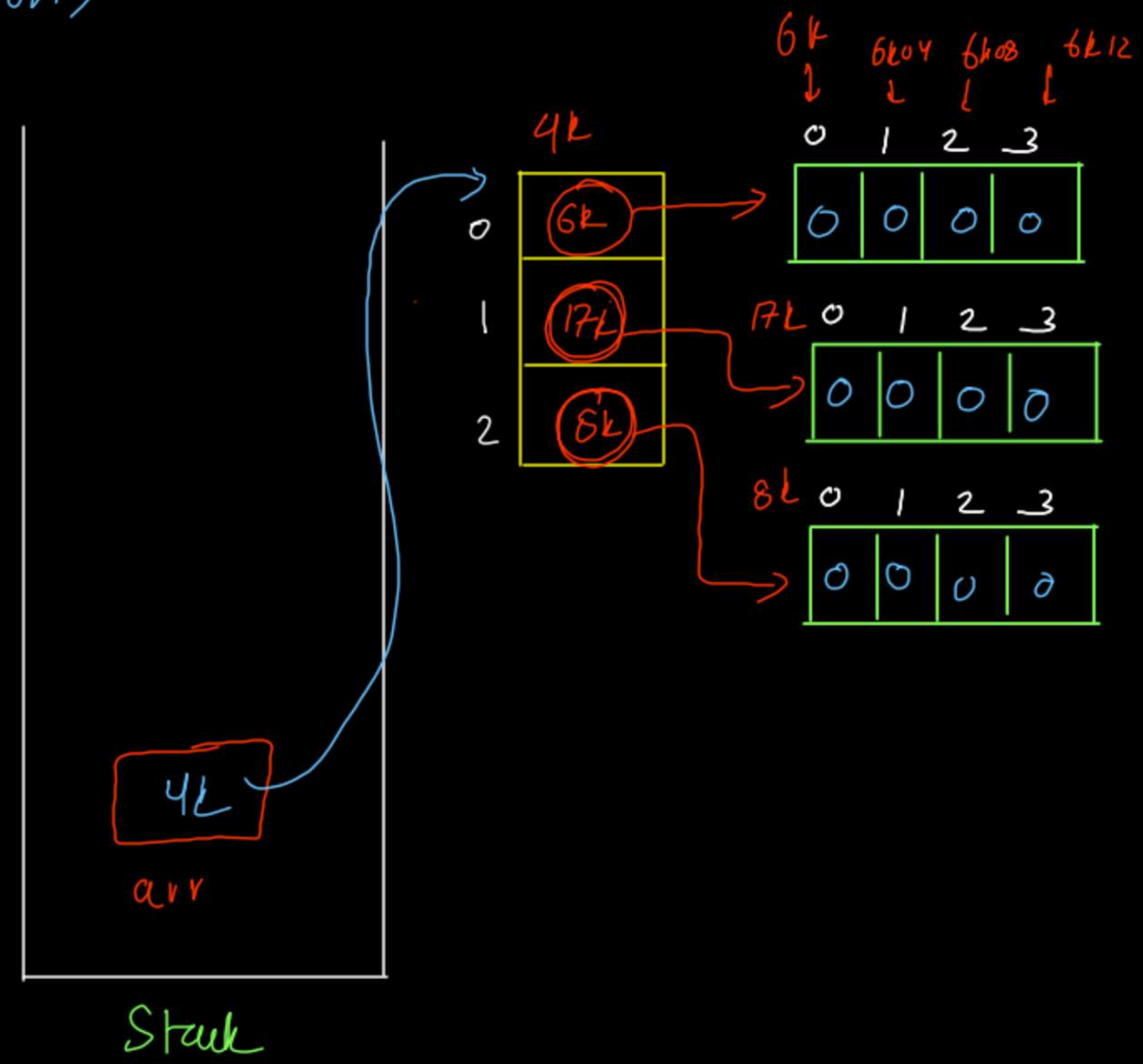
row col

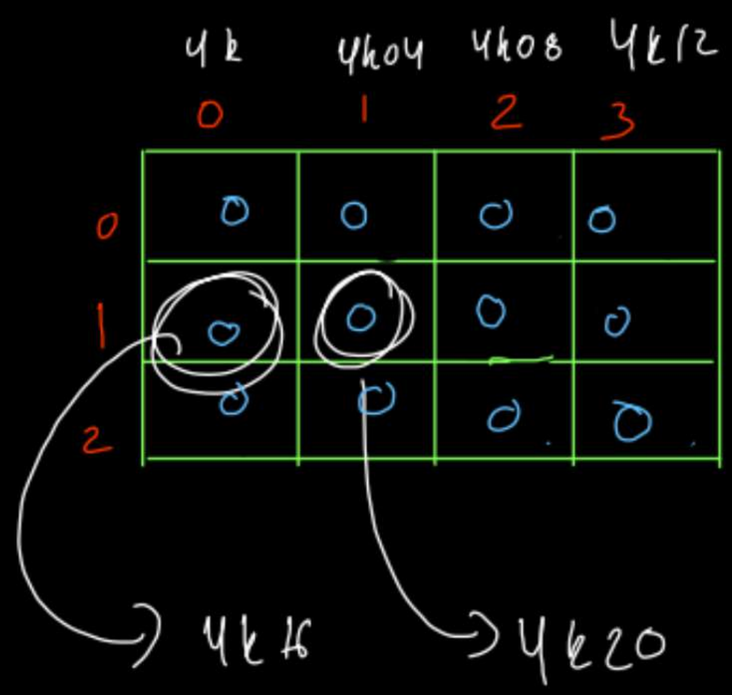
	0	1	2	3
0	0	0	0	0
1	0	0	0	0
2	0	0	0	0

Memory Mapping (Initialization)

```
int [][] arr;   ①
arr = new int [3][4];
```

row col



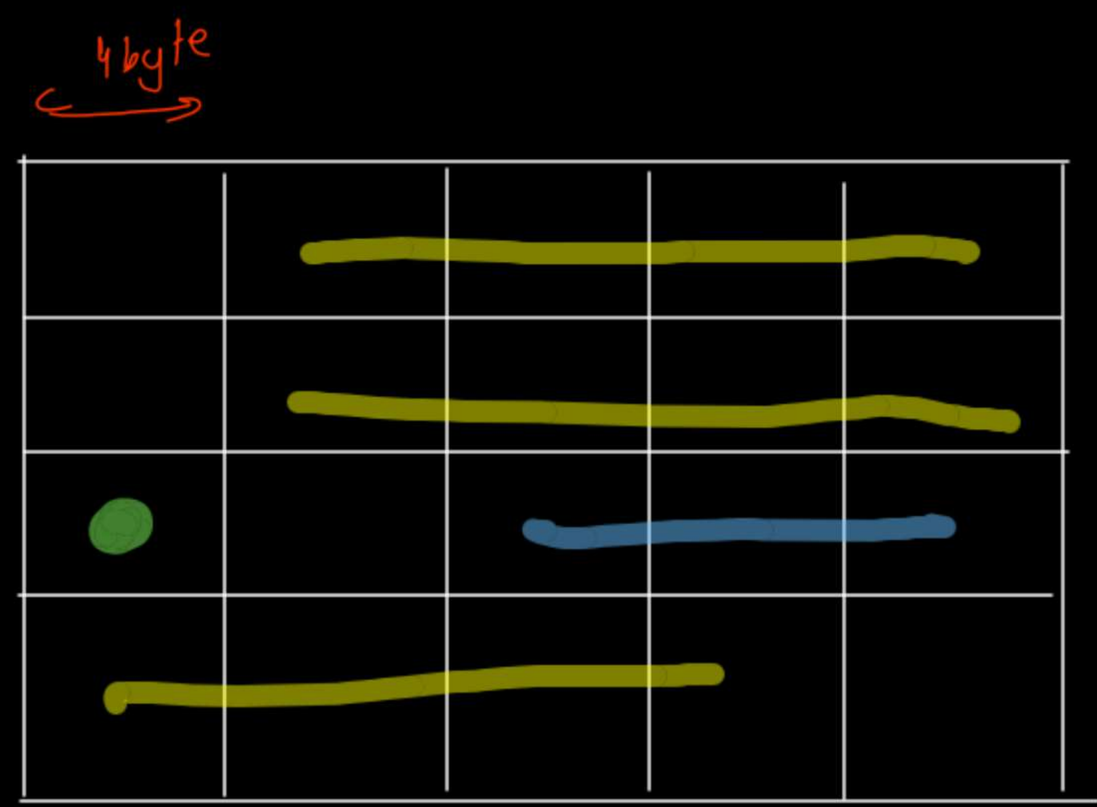


→ 1-D Array of Size 12

$4 \times 12 = 48$ bytes

C++ Row-Major Mapping

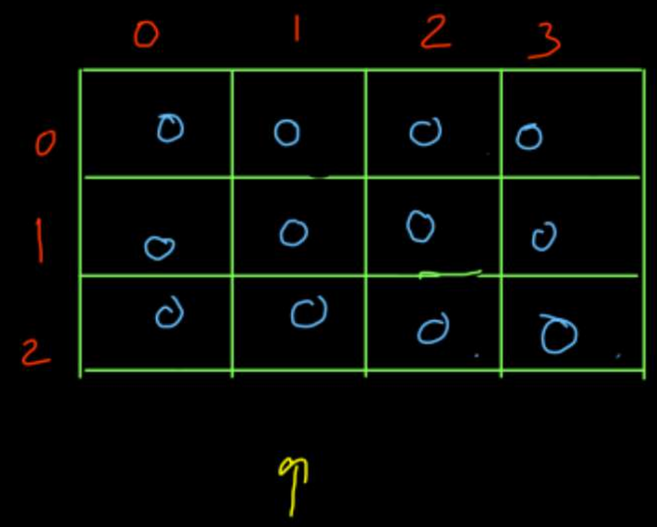
fragmentation



In C++ fragmentation is more likely to happen as there is contiguos memory allocation throughout the matrix.

In Java, it is rare to happen

Length Property



arr.length = gives the length of array of rows

(no of rows)

arr[0].length = gives the no of cols.

Direct Access

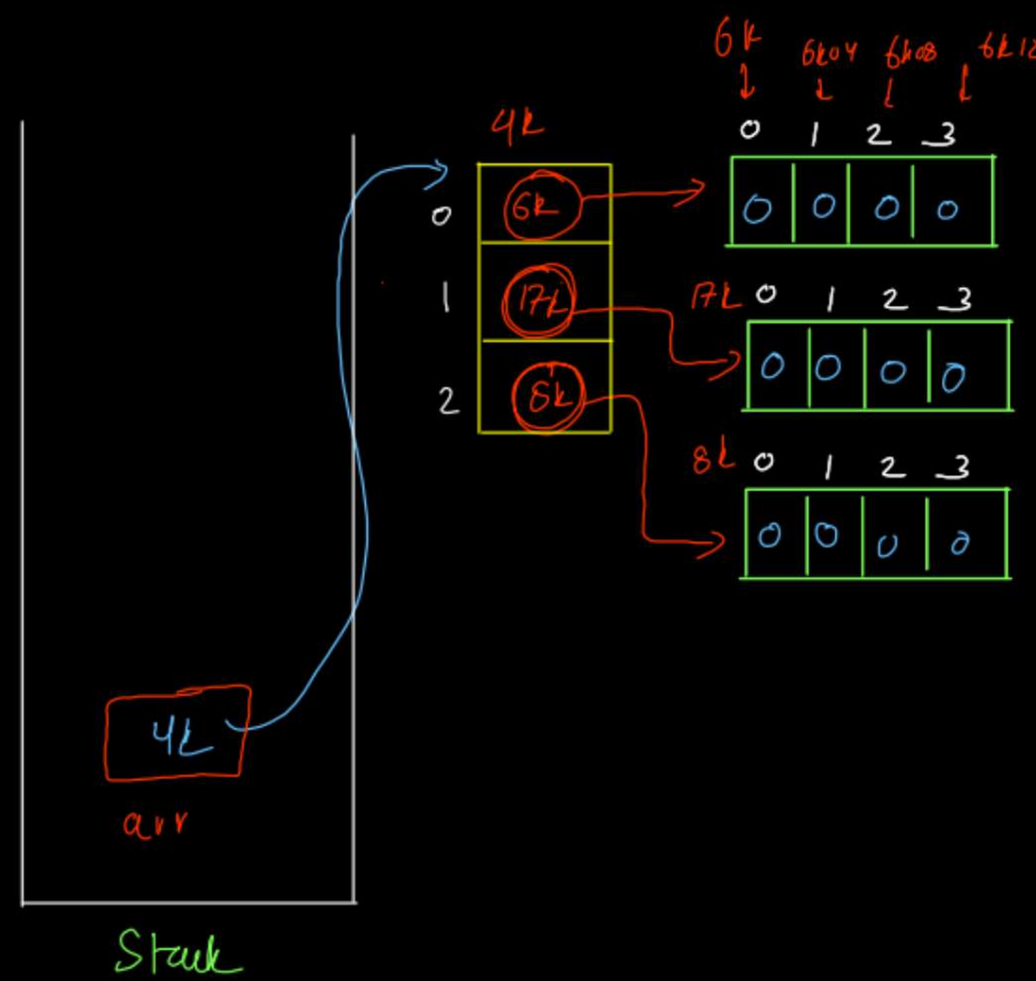
① int [][] arr;

arr = new int [3][4];

row col

arr[1][2] = ?

LOC = base + DT * index
add



Traverse

	0	1	2	3
0	0	0	0	0
1	0	0	0	0
2	0	0	0	0

```java
public static void main(String[] args) throws Exception {
    // write your code here
    Scanner scn = new Scanner(System.in);
    int row = scn.nextInt();
    int col = scn.nextInt();

    //declaration + initialization
    int[][] mat = new int[row][col];

    for(int i=0;i<mat.length;i++) {
        for(int j=0;j<mat[0].length;j++) {
            mat[i][j] = scn.nextInt();
        }
    }

    for(int i=0;i<mat.length;i++) {
        for(int j=0;j<mat[0].length;j++) {
            System.out.print(mat[i][j] + " ");
        }
        System.out.println();
    }
}

}
```

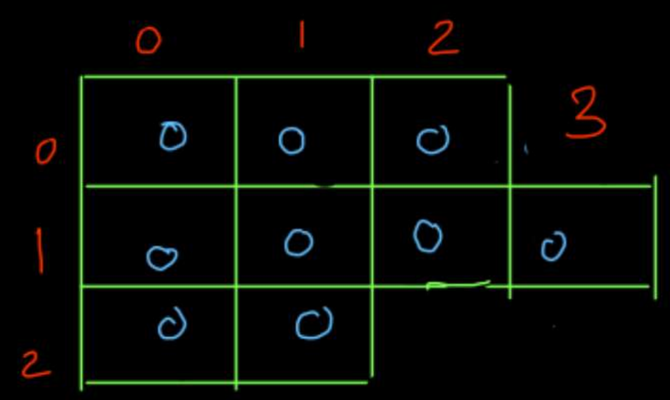
Lecture: 13

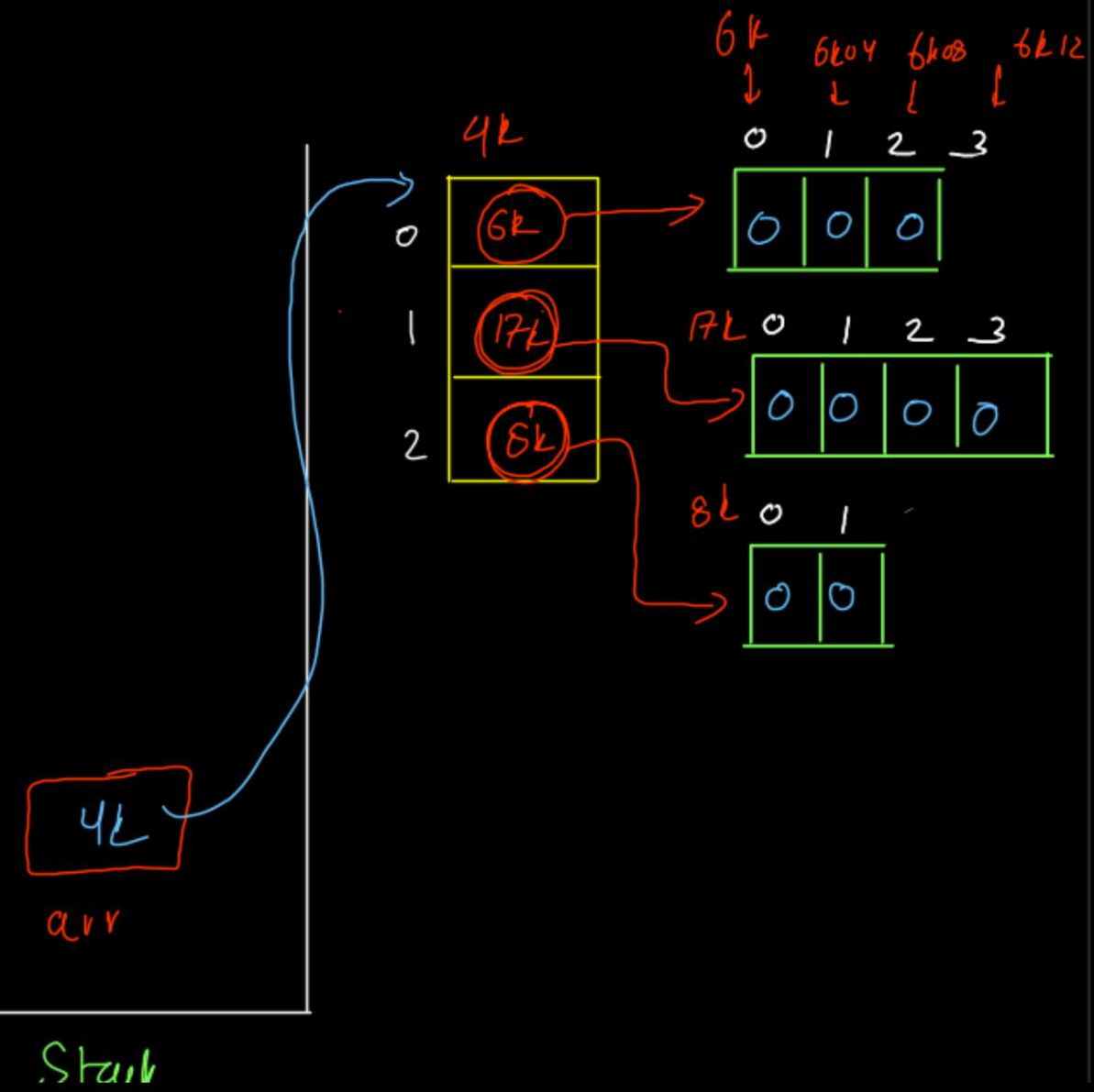
Jagged Array

2-D Array as a function parameter & return type (Shallow & Deep copy)

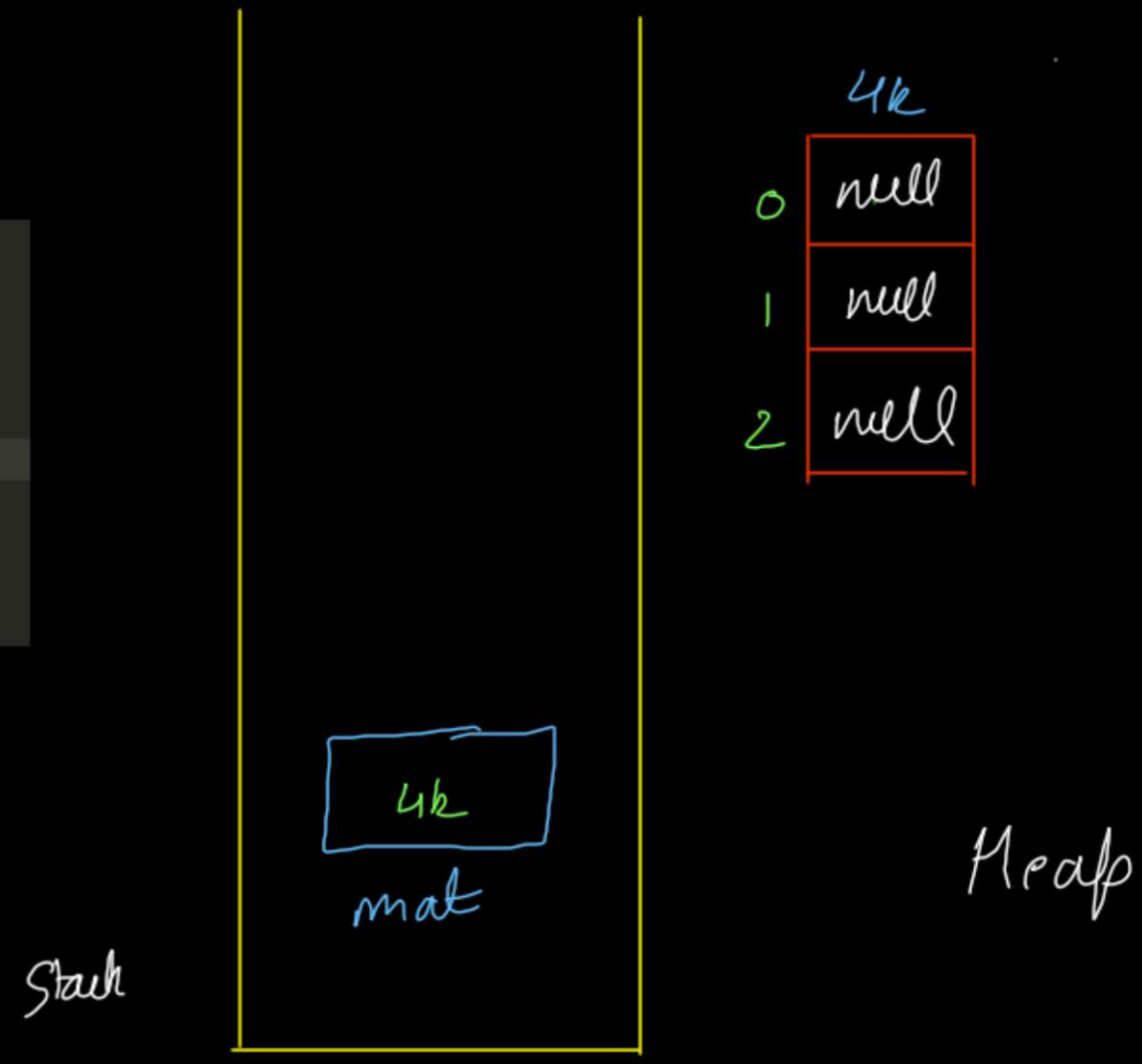
1. Matrix Multiplication
2. Rotate Image

Jagged Array





```java
import java.util.*;

public class Main {
    public static void main(String[] args) {
        int[][] mat = new int[3][];

    }
}

```



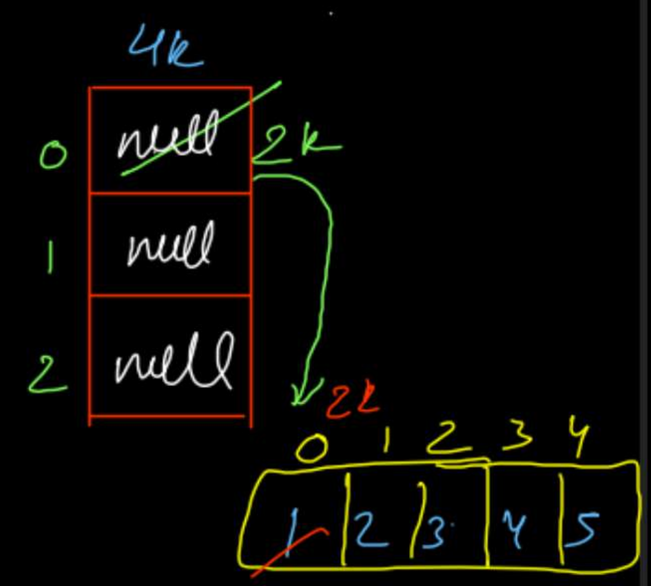
$LOC = BA + DT * index$

$= BA$

```java
import java.util.*;

public class Main {
    public static void main(String[] args) {
        int[][] mat = new int[3][];
        int[] arr1 = {1,2,3,4,5};
        mat[0] = arr1;

    }
}
```

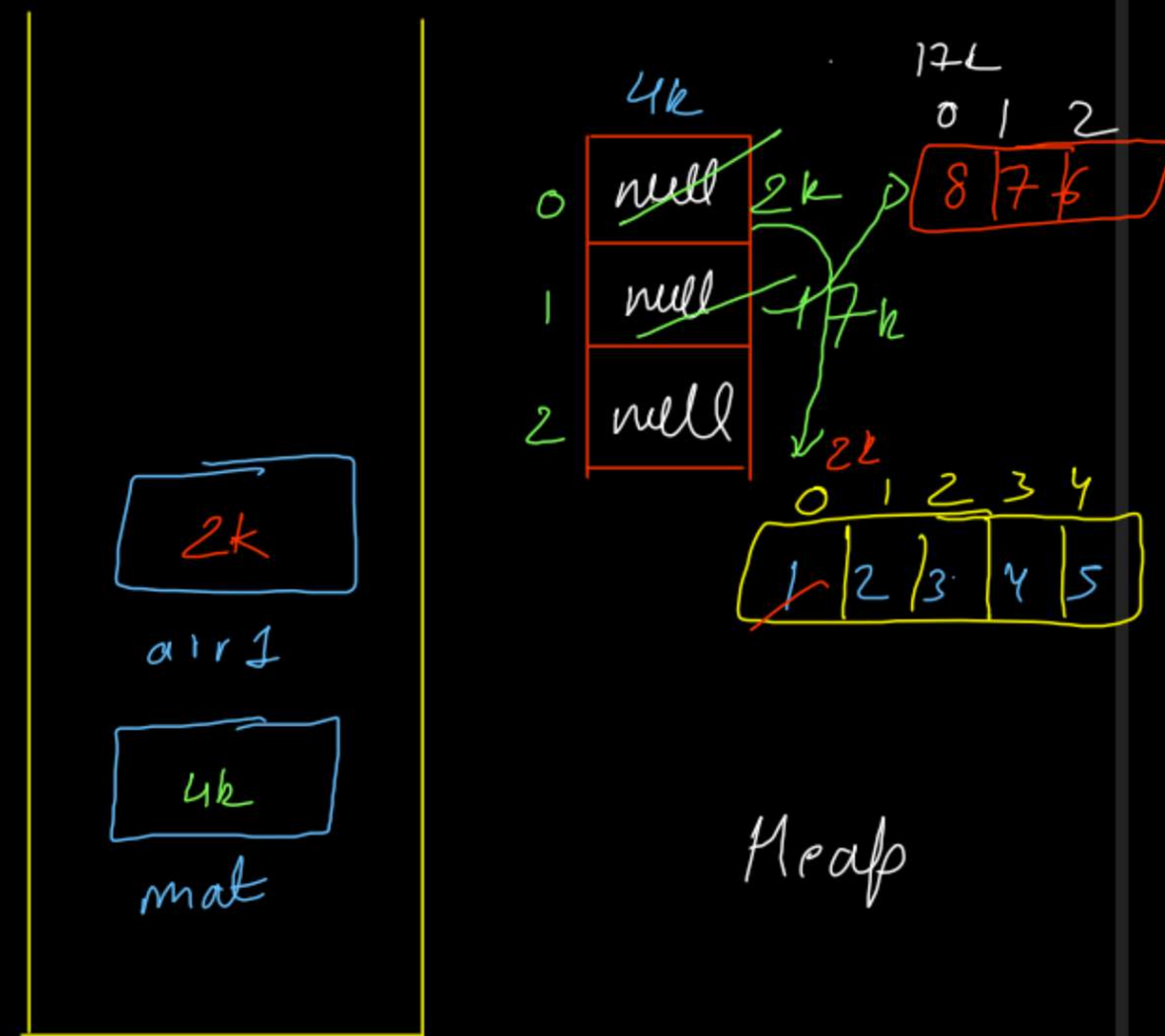
mat[0] = arr1

arr1[0] = 10



Heap

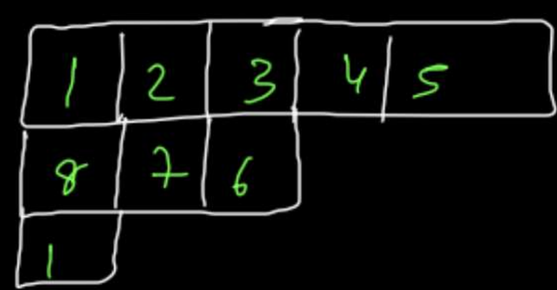
```java
import java.util.*;

public class Main {
    public static void main(String[] args) {
        int[][] mat = new int[3][];
        int[] arr1 = {1,2,3,4,5};
        mat[0] = arr1;

        mat[1] = new int[]{8,7,6};
    }
}
```

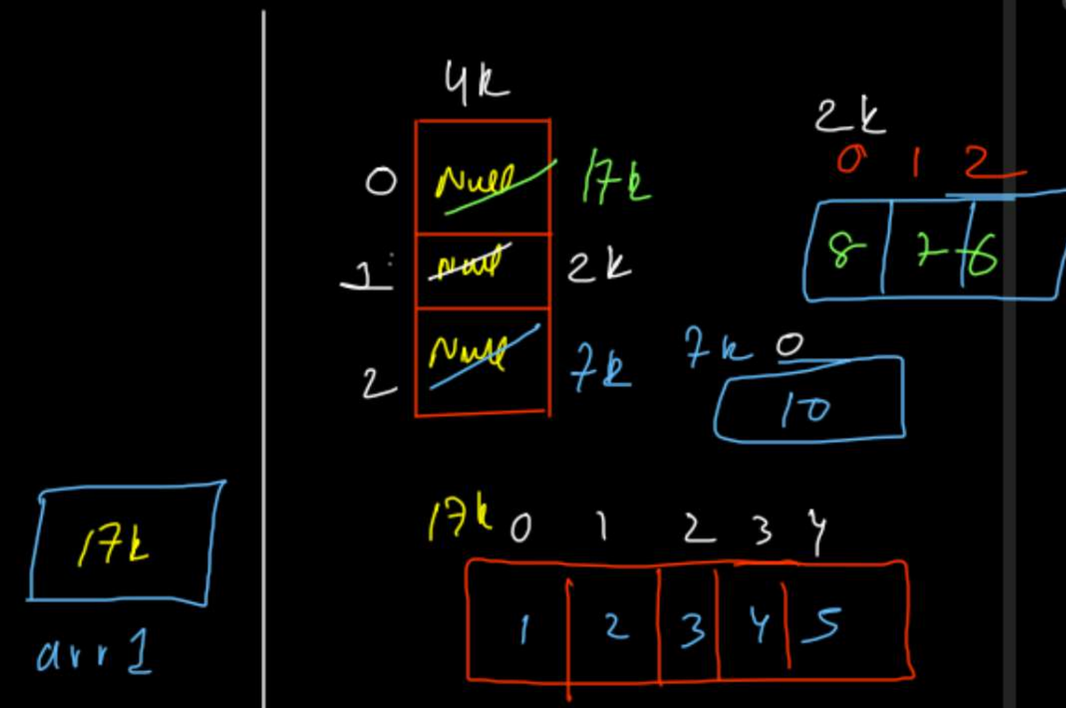


```java
import java.util.*;

public class Main {
    public static void main(String[] args) {
        int[][] mat = new int[3][];
        int[] arr1 = {1,2,3,4,5};
        mat[0] = arr1;

        mat[1] = {8,7,6};
    }
}
```

This is syntax error.

```java
import java.util.*;

public class Main {
    public static void main(String[] args) {
        int[][] mat = new int[3][];
        int[] arr1 = {1,2,3,4,5};
        mat[0] = arr1;
        mat[1] = new int[]{8,7,6};
        mat[2] = new int[]{10};


    }
}
```

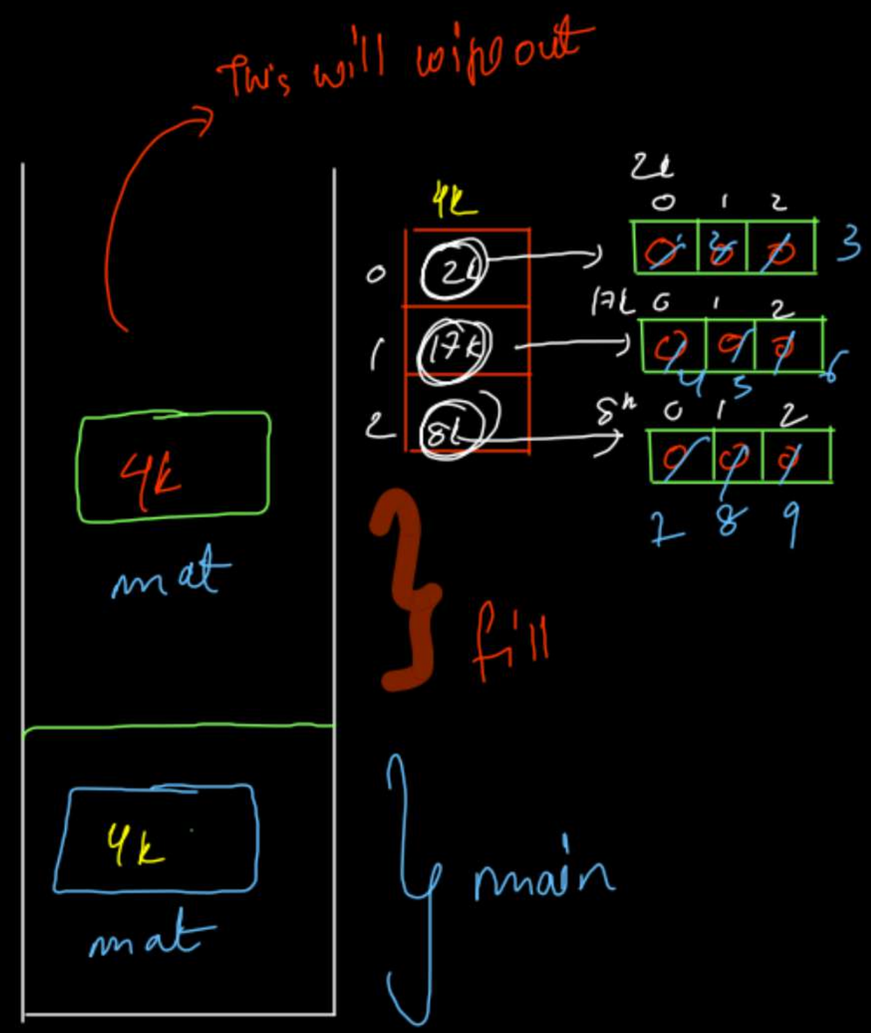




```java
import java.util.*;

public class Main {
    public static void main(String[] args) {
        int[][] mat = new int[3][];
        int[] arr1 = {1,2,3,4,5};
        mat[0] = arr1;
        mat[1] = new int[]{8,7,6};
        mat[2] = new int[]{10};

        for(int i=0;i<mat.length;i++) {
            for(int j=0;j<mat[i].length;j++) {
                System.out.print(mat[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

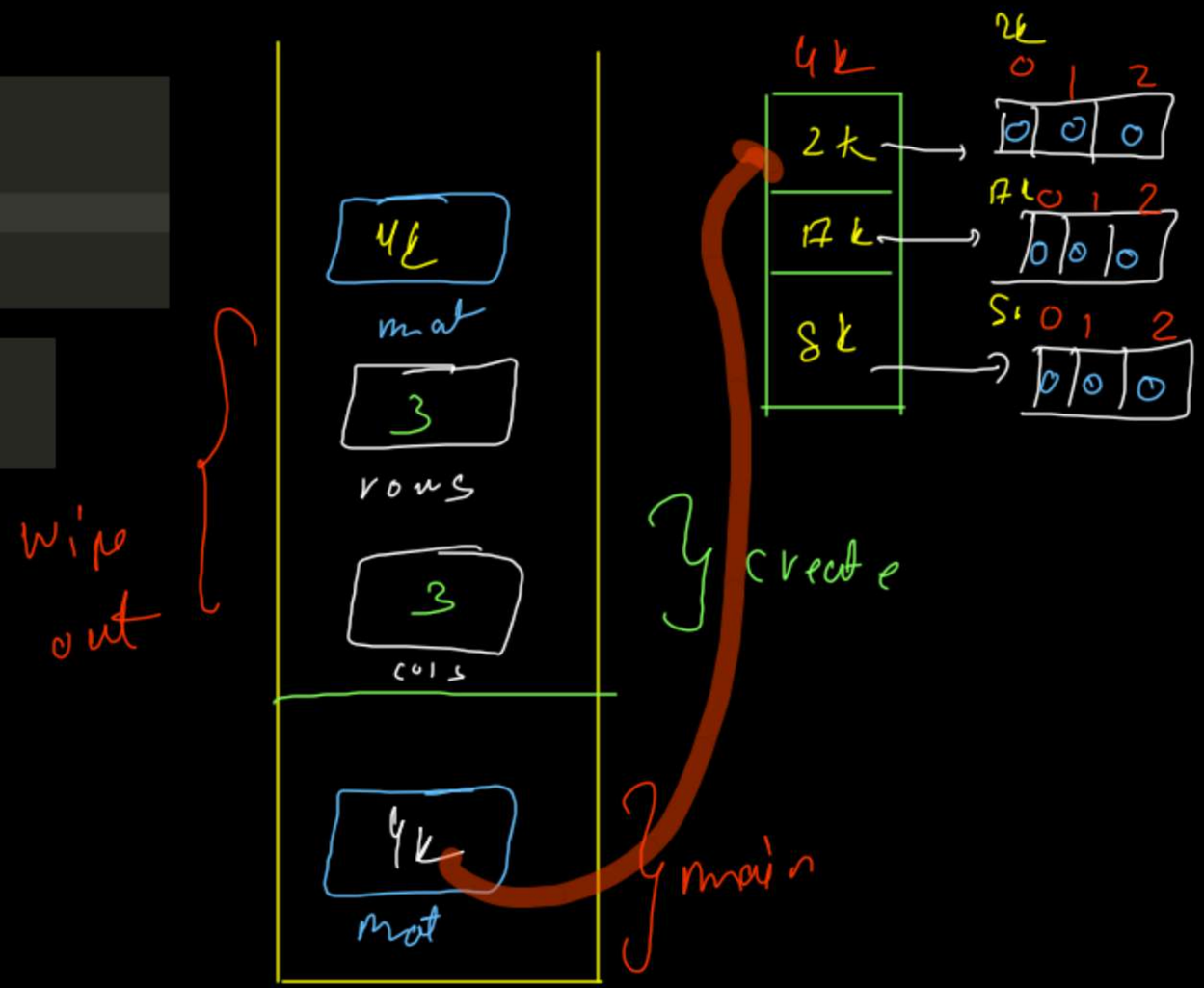
2-D Array as a Method Parameter

```java
public static void fill(int[][] mat) {

    int count = 1;

    for(int i=0;i<mat.length;i++) {
        for(int j=0;j<mat[0].length;j++) {
            mat[i][j] = count++;
        }
    }
}

public static void main(String[] args) {
    int[][] mat = new int[3][3];

    fill(mat);

    for(int i=0;i<mat.length;i++) {
        for(int j=0;j<mat[0].length;j++) {
            System.out.print(mat[i][j] + " ");
        }
    }
}
```



```java
public static int[][] create(int rows,int cols) {
    int[][] mat = new int[rows][cols];
    return mat;
}
```

```java
public static void main(String[] args) {
    int[][] mat = create(3,3);
```



Matrix Multiplication

Easy

Prev Next

1. You are given a number n1, representing the number of rows of 1st matrix.
2. You are given a number m1, representing the number of columns of 1st matrix.
3. You are given n1*m1 numbers, representing elements of 2d array a1.
4. You are given a number n2, representing the number of rows of 2nd matrix.
5. You are given a number m2, representing the number of columns of 2nd matrix.
6. You are given n2*m2 numbers, representing elements of 2d array a2.
7. If the two arrays representing two matrices of dimensions n1 * m1 and n2 * m2 can be multiplied, display the contents of prd array as specified in output Format.
8. If the two arrays can't be multiplied, print "Invalid input".

$$\begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 1 & 2 & 1 \\ 1 & 2 & 0 & 1 \end{array}_{2\times 3} \qquad \begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 2 \\ 2 & 1 & 2 & 0 \end{array}_{3\times 3}$$

Matrix multiplication is not commutative

$$A * B \neq B * A$$

$$\begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 1 & 2 & 1 \\ 1 & 2 & 0 & 1 \end{array}_{2\times 3} \qquad \begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 2 \\ 2 & 1 & 2 & 0 \end{array}_{3\times 3}$$

r_1 c_1 $\qquad$ r_2 c_2

for matrix multiplication, it is a rule that $c_1 = r_2$.

$$\begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 1 & 2 & 1 \\ 1 & 2 & 0 & 1 \end{array}_{2\times 3} \qquad \begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 2 \\ 2 & 1 & 2 & 0 \end{array}_{3\times 3} = \begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 2 & 4 & 5 \\ 1 & 3 & 2 & 2 \end{array}_{2\times 3}$$

$$\begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & a_{00} & a_{01} & a_{02} \\ 1 & a_{10} & a_{11} & a_{12} \end{array}_{2\times 3} \times \begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & b_{00} & b_{01} & b_{02} \\ 1 & b_{10} & b_{11} & b_{12} \\ 2 & b_{20} & b_{21} & b_{22} \end{array}_{3\times 3} = \begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & c_{00} & c_{01} & c_{02} \\ 1 & c_{10} & c_{11} & c_{12} \end{array}_{2\times 3}$$

A $\qquad$ B $\qquad$ $r_1\ c_2$

$$c_{01} = a_{00} * b_{01} + a_{01} * b_{11} + a_{02} * b_{21}$$

$$\begin{array}{cc} i & j \end{array} \qquad \begin{array}{cc} i & k \end{array} \begin{array}{cc} k & j \end{array} \qquad \begin{array}{cc} i & k \end{array} \begin{array}{cc} k & j \end{array} \qquad \begin{array}{cc} i & k \end{array} \begin{array}{cc} k & j \end{array}$$

$$c_{ij} = \sum a_{ik} * b_{kj}$$

```java
if(c1 != r2) {
    System.out.println("Invalid input");
    return;
}

int[][] c = new int[r1][c2];

int commonDimension = c1; //or r2

for(int i=0;i<c.length;i++) {
    for(int j=0;j<c[0].length;j++) {
        for(int k=0;k<commonDimension;k++) {
            c[i][j] += a[i][k] * b[k][j];
        }
    }
}

for(int i=0;i<c.length;i++) {
    for(int j=0;j<c[0].length;j++) {
        System.out.print(c[i][j] + " ");
    }
    System.out.println();
}
```

```
class Grd

    public static void multiply(int A[][], int B[][], int C[][], int N)
      {

          int commonDimension = A[0].length;

         //add code here.
          for(int i=0;i<C.length;i++) {
            for(int j=0;j<C[0].length;j++) {
                for(int k=0;k<commonDimension;k++) {
                    C[i][j] += A[i][k] * B[k][j];
                }
             }
  }

      }
```

$O(N^3)$

48. Rotate Image

Medium 12097 575 Add to List Share

You are given an `n x n` 2D `matrix` representing an image, rotate the image by **90** degrees (clockwise).

You have to rotate the image **in-place**, which means you have to modify the input 2D matrix directly. **DO NOT** allocate another 2D matrix and do the rotation.

Example 1:

1	2	3
4	5	6
7	8	9

⇨

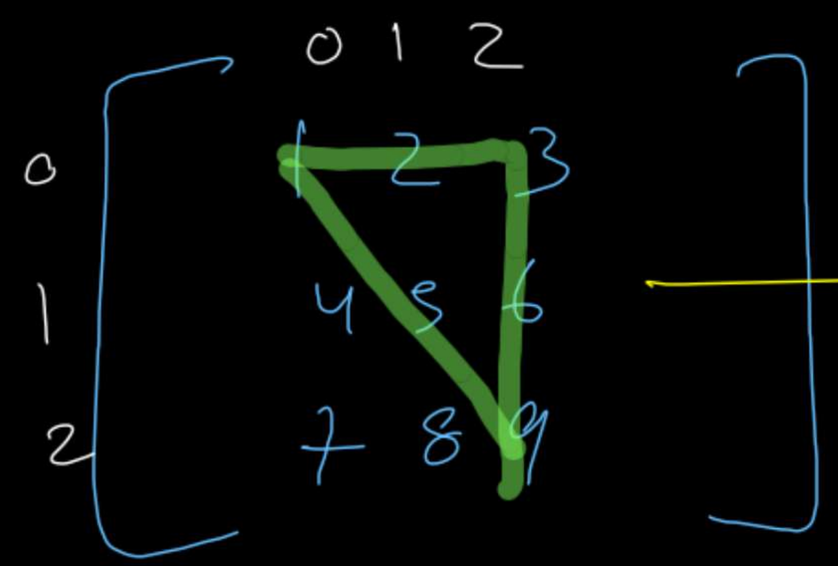
7	4	1
8	5	2
9	6	3

$$\begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 1 & 2 & 3 \\ 1 & 4 & 5 & 6 \\ 2 & 7 & 8 & 9 \end{array} \xrightarrow{\text{rot}} \begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 7 & 4 & 1 \\ 1 & 8 & 5 & 2 \\ 2 & 9 & 6 & 3 \end{array}$$

$$\xrightarrow[\text{1}]{\text{transpose}} \begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 1 & 4 & 7 \\ 1 & 2 & 5 & 8 \\ 2 & 3 & 6 & 9 \end{array} \xrightarrow[\text{2}]{\text{reverse each row}} \begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 7 & 4 & 1 \\ 1 & 8 & 5 & 2 \\ 2 & 9 & 6 & 3 \end{array}$$

Transpose

↳ swapping of arr[i][j] & arr[j][i]



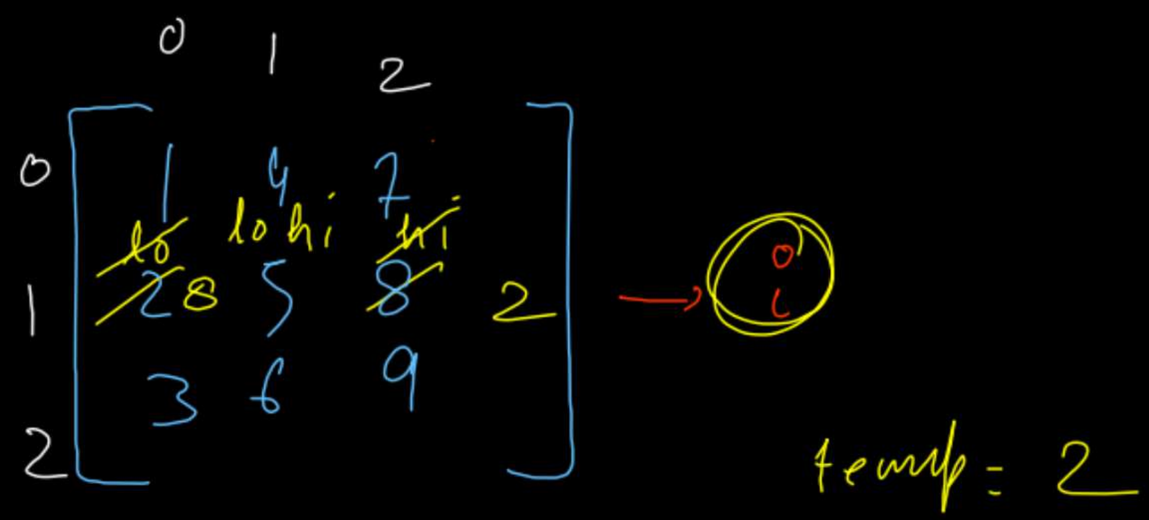
col = row ; col < n

→ Traverse only in Upper Triangular Half

$\boxed{col \geq row}$

Reverse

reverse(mat, i)




```java
public void transpose(int[][] mat) {
    //traverse only in upper triangle
    for(int i=0;i<mat.length;i++) {
        for(int j=i;j<mat[0].length;j++) {
            int temp = mat[i][j];
            mat[i][j] = mat[j][i];
            mat[j][i] = temp;
        }
    }
}
```

```java
public void rotate(int[][] matrix) {
    transpose(matrix);

    for(int i=0;i<matrix.length;i++) {
        reverseRow(matrix,i);
    }
}
```

```java
public void reverseRow(int[][] matrix,int i) {
    int lo = 0;
    int hi = matrix[0].length -1;

    while(lo < hi) {
        int temp = matrix[i][lo];
        matrix[i][lo] = matrix[i][hi];
        matrix[i][hi] = temp;
        lo++;
        hi--;
    }
}
```

Lecture - 14

Exit Point

Spiral Matrix

Saddle Point

Search in a 2-D Matrix - 1

Search in a 2-D Matrix - 2

Ring Rotate

Exit Point of a Matrix

Exit Point in a Matrix

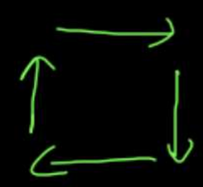
Easy Accuracy: **42.96%** Submissions: **1976** Points: **2**

Given a matrix of size n x m with 0's and 1's , you enter the matrix at cell (0,0) in left to right direction. whenever you encounter a 0 you retain in same direction , if you encounter a 1's you have to change direction to right of current direction and change that 1 value to 0, you have to find out from which index you will leave the matrix at the end.

Start point

$$\begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 0 & \cancel{1}\,0 & 0 \\ 1 & 0 & \cancel{1}\,0 & 1 \\ 2 & 0 & 0 & 0 \end{array}$$

Starting direction: $\longrightarrow$

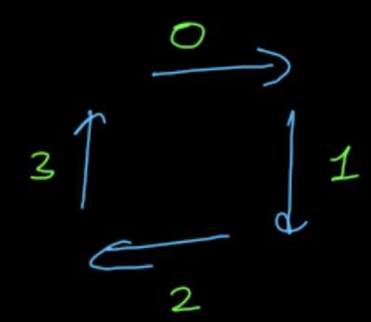


$$\begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 0 & \cancel{1}\,0 & 0 \\ 1 & 0 & \cancel{1}\,0 & 1 \\ 2 & 0 & 0 & 0 \end{array}$$

Initially dir = 0

$i \rightarrow$ row

$j \rightarrow$ col



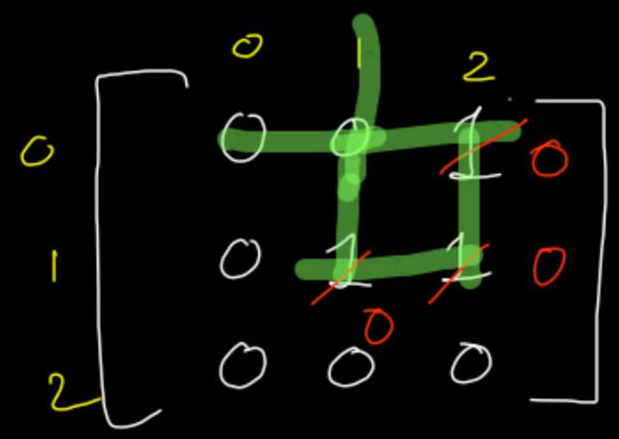
dir = 0 $\rightarrow$ j++;

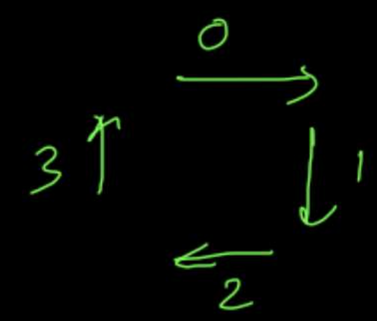
dir = 1 $\rightarrow$ i++;

dir = 2 $\rightarrow$ j--;

dir = 3 $\rightarrow$ i--;

dir = $\cancel{0}$ $\cancel{1}$ 2





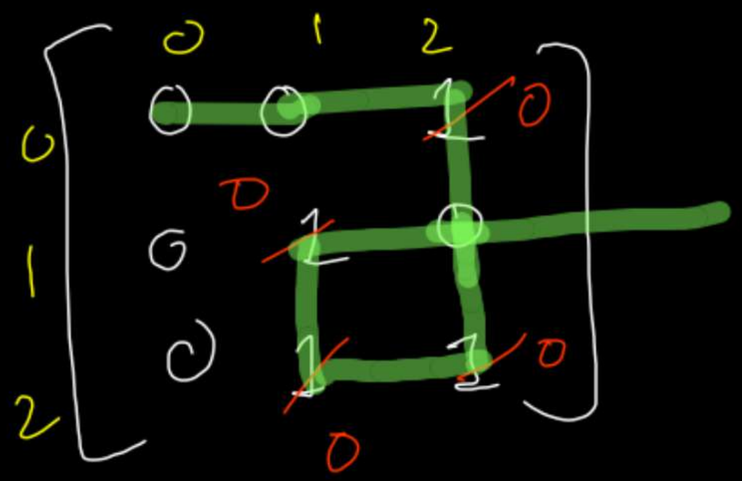
$dir = (dir + 1) \% 4 \rightarrow 0, 1, 2, 3$

$dir = 0 \rightarrow j{+}{+};$

$dir = 1 \rightarrow i{+}{+};$

$dir = 2 \rightarrow j{-}{-};$

$dir = 3 \rightarrow i{-}{-};$



$dir = \cancel{0}\ \cancel{1}\ \cancel{2}\ 3$

$dir = (dir+1) \% 4$

$(1+1) \% 4 = 2 \% 4 = 2$

$dir = (2+1) \% 4 = 3 \% 4 = 3$

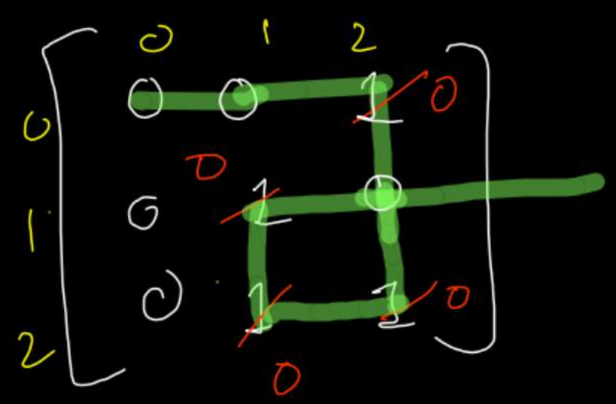
$dir = (3+1) \% 4 = 4 \% 4 = 0$

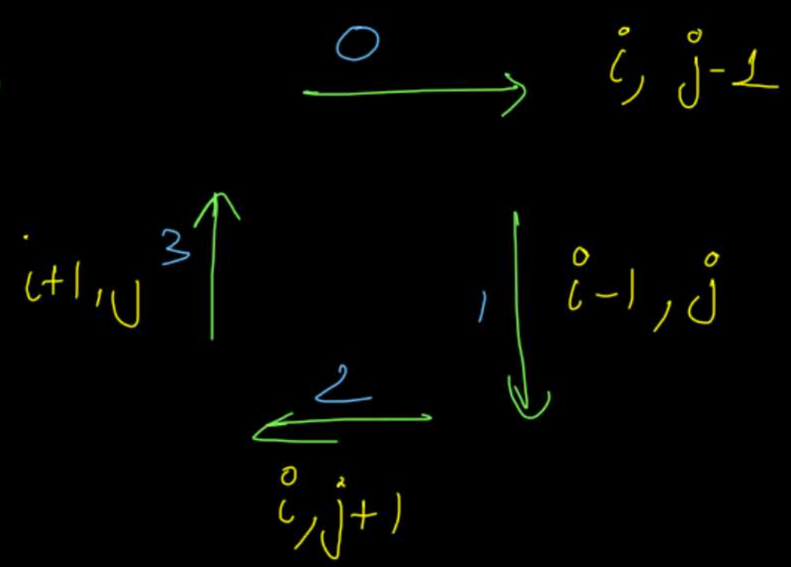
$dir = 0 \rightarrow j{+}{+};$

$dir = 1 \rightarrow i{+}{+};$

$dir = 2 \rightarrow j{-}{-};$

$dir = 3 \rightarrow i{-}{-};$





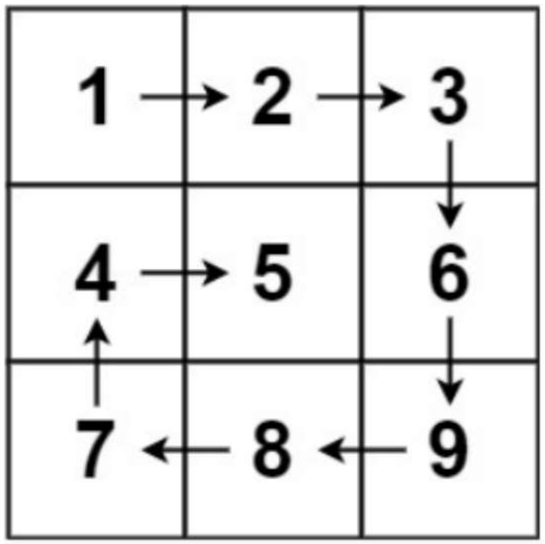
```java
public int[] FindExitPoint(int[][] matrix)
{
    // code here
    int[] res = new int[2];
    int i = 0;
    int j = 0;
    int dir = 0;

    while(i >= 0 && i < matrix.length && j >=0 && j < matrix[0].length) {
        if(matrix[i][j] == 1) {
            dir = (dir + 1) % 4;
            matrix[i][j] = 0;
        }

        if(dir == 0) {
            j++;
        } else if(dir == 1) {
            i++;
        } else if(dir == 2) {
            j--;
        } else {
            i--;
        }
    }

    //you have exited the matrix
    if(dir == 0) {
        j--;
    } else if(dir == 1) {
        i--;
    } else if(dir == 2) {
        j++;
    } else {
        i++;
    }

    res[0] = i;
    res[1] = j;

    return res;
}
```

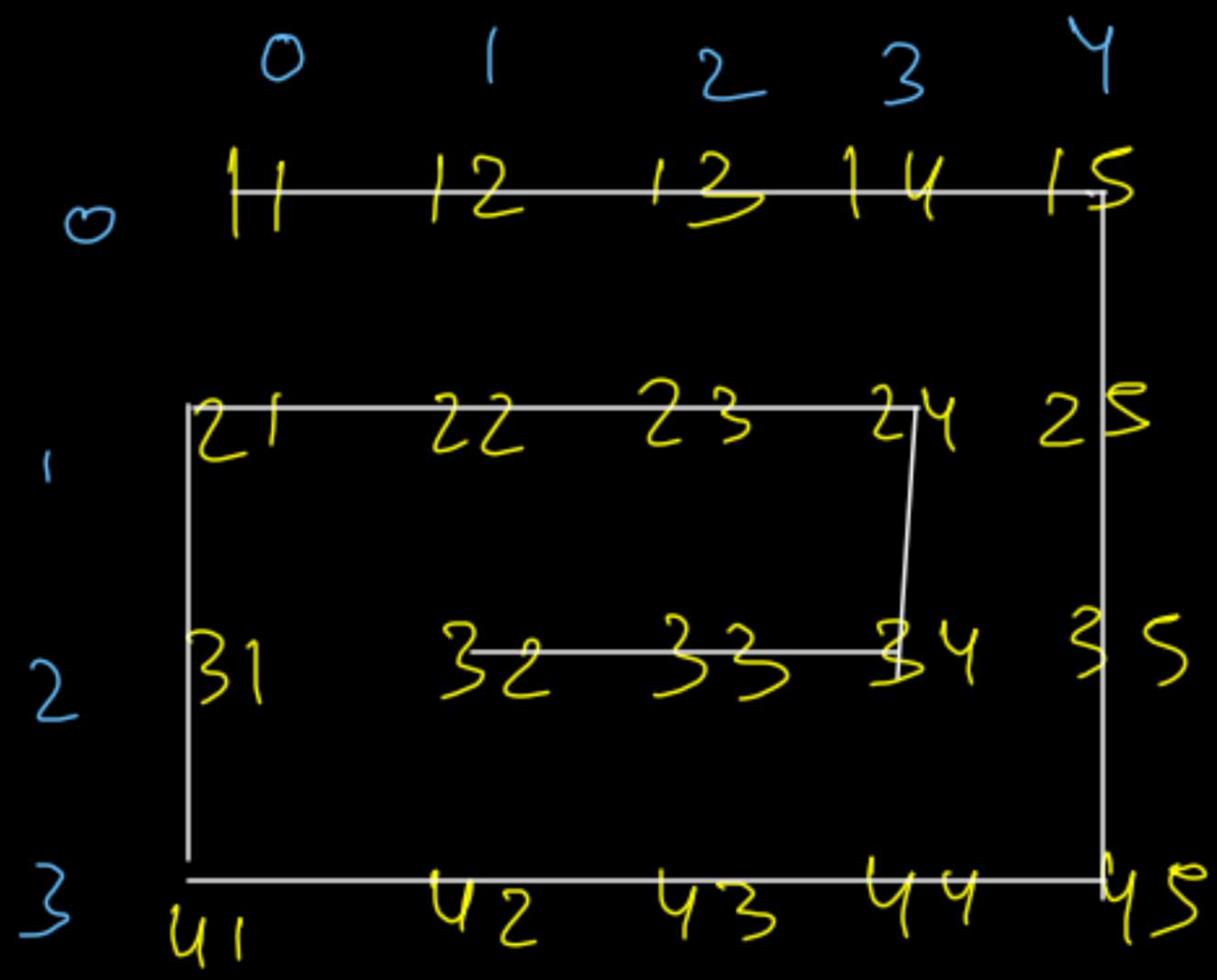
54. Spiral Matrix

Medium 8885 914 Add to List Share

Given an `m x n` `matrix`, return *all elements of the* `matrix` *in spiral order.*

Example 1:




0 1 2 3 4
0
1
2
3
11 12 13 14 15
21 22 23 24 25
31 32 33 34 35
41 42 43 44 45

	0	1	2	3	4
0	11	12	13	14	15
1	21	22	23	24	25
2	31	32	33	34	35
3	41	42	43	44	45

fr = 0 lr = mat.length - 1

fc = 0 lc = mat[0].length - 1

1 # first row

j = fc; j ≤ lc; j++

fr++;

2 # Last col

i = fr; i ≤ lr; i++

lc--;

3 # Last row

j = lc; j ≥ fc; j--

lr--;

4 # first col

i = lr; i ≥ fr; i--

fc++

```java
    int rows = matrix.length;
    int cols = matrix[0].length;
    int fr = 0, fc = 0, lr = rows-1, lc = cols -1;
    int count = 0;

    List<Integer> list = new ArrayList<>();
```

```java
    while(count != rows * cols) {
        //first row
        for(int j=fc;j<=lc;j++) {
            list.add(matrix[fr][j]);
            count++;
        }
        fr++;

        if(count == rows*cols) break;

        //last col
        for(int i=fr;i<=lr;i++) {
            list.add(matrix[i][lc]);
            count++;
        }
        lc--;

        if(count == rows*cols) break;

        //last row
        for(int j=lc;j>=fc;j--) {
            list.add(matrix[lr][j]);
            count++;
        }
        lr--;

        if(count == rows*cols) break;
```

```java
        if(count == rows*cols) break;

        //first col
        for(int i=lr;i>=fr;i--) {
            list.add(matrix[i][fc]);
            count++;
        }
        fc++;

        if(count == rows*cols) break;
    }

    return list;
```

240. Search a 2D Matrix II

Medium 9221 153 Add to List Share

Write an efficient algorithm that searches for a value `target` in an `m x n` integer matrix `matrix`. This matrix has the following properties:

- Integers in each row are sorted in ascending from left to right.
- Integers in each column are sorted in ascending from top to bottom.

Search in a Row-wise & Column Wise Sorted Matrix

Example 1:

1	4	7	11	15
2	5	8	12	19
3	6	9	16	22
10	13	14	17	24
18	21	23	26	30

Approach - 1

{Brute force}

Search in entire matrix

$O(M*N)$

Approach - 2

Apply Binary Search on each row/col.

$O(M*\log(N))$ or $O(N\log(M))$

```java
public boolean binSearch(int[][] matrix,int i,int target) {
    int lo = 0;
    int hi = matrix[0].length -1;

    while(lo <= hi) {
        int mid = lo + (hi-lo)/2;

        if(matrix[i][mid] == target) return true;
        else if(matrix[i][mid] < target) lo = mid + 1;
        else hi = mid -1;
    }

    return false;
}
```

```java
public boolean searchMatrix(int[][] matrix, int target) {

    for(int i=0;i<matrix.length;i++) {
        if(target >= matrix[i][0] && target <= matrix[i][matrix[0].length-1]) {
            boolean val = binSearch(matrix,i,target);
            if(val == true) return true;
        }
    }

    return false;
}
```

Approach-3 {Staircase Search}

$O(N+M)$

Example 1:

1	4	7	11	15
2	5	8	12	19
3	6	9	16	22
10	13	14	17	24
18	21	23	26	30

target = 20

while (i < matrix.length

&& j >= 0)

{

}

```java
class Solution {
    public boolean searchMatrix(int[][] matrix, int target) {
        int i = 0;
        int j = matrix[0].length -1;

        while(i < matrix.length && j >= 0) {
            if(matrix[i][j] == target) return true;
            if(matrix[i][j] < target) i++;
            else j--;
        }

        return false;
    }
}
```

74. Search a 2D Matrix

Medium 9648 304 Add to List Share

Write an efficient algorithm that searches for a value `target` in an `m x n` integer matrix `matrix`. This matrix has the following properties:

- Integers in each row are sorted from left to right.
- The first integer of each row is greater than the last integer of the previous row.

Search in a Sorted Matrix

Example 1:

	0	1	2	3
0	1	3	5	7
1	10	11	16	20
2	23	30	34	60

target = 30

mid lo hi mid lo hi

0	1	2	3	4	5	6	7	8	9	10	11
1,	3,	5,	7,	10,	11	16,	20,	23,	30,	34,	60

lo = 6

hi = 11

mid = $\frac{6+11}{2}$ = 8

rIdx = 1

cIdx = 1

rowIdx = idx/cols ;

colIdx = idx%cols;

if(mat[rIdx][cIdx] == target)

lo = mid + 1

1-D Array Idx = idx

Row Idx = idx / cols;

Col Idx = idx % cols;

```java
class Solution {
    public boolean searchMatrix(int[][] matrix, int target) {
        int lo = 0;
        int hi = matrix.length * matrix[0].length -1;

        while(lo <= hi) {
            int mid = lo + (hi-lo)/2;

            int rIdx = mid / matrix[0].length;
            int cIdx = mid % matrix[0].length;

            if(matrix[rIdx][cIdx] == target) return true;
            else if(matrix[rIdx][cIdx] < target) lo = mid + 1;
            else hi = mid -1;
        }

        return false;
    }
}
```

target = 30

	0	1	2	3
0	1	3	5	7
1	10	11	16	20
2	23	30	34	60

mid lo hi

0	1	2	3	4	5	6	7	8	9	10	11
1,	3,	5,	7,	10,	11,	16,	20,	23,	30,	34,	60

arr[mid] == target

rowIdx = idx/cols;

colIdx = idx % cols;

5/4 → 1 → row

5%4 → 1 → col

Saddle Price

Easy

Prev Next

1. You are given a square matrix of size 'n'. You are given n*n elements of the square matrix.
2. You are required to find the saddle price of the given matrix and print the saddle price.
3. The saddle price is defined as the least price in the row but the maximum price in the column of the matrix.

$$\begin{array}{c|ccc} & 1 & 2 & 3 \\ \hline 1 & a_{11} & a_{12} & a_{13} \\ 2 & a_{21} & a_{22} & a_{23} \\ 3 & a_{31} & a_{32} & a_{33} \end{array}$$

$a_{11} \rightarrow$ min value in row 1

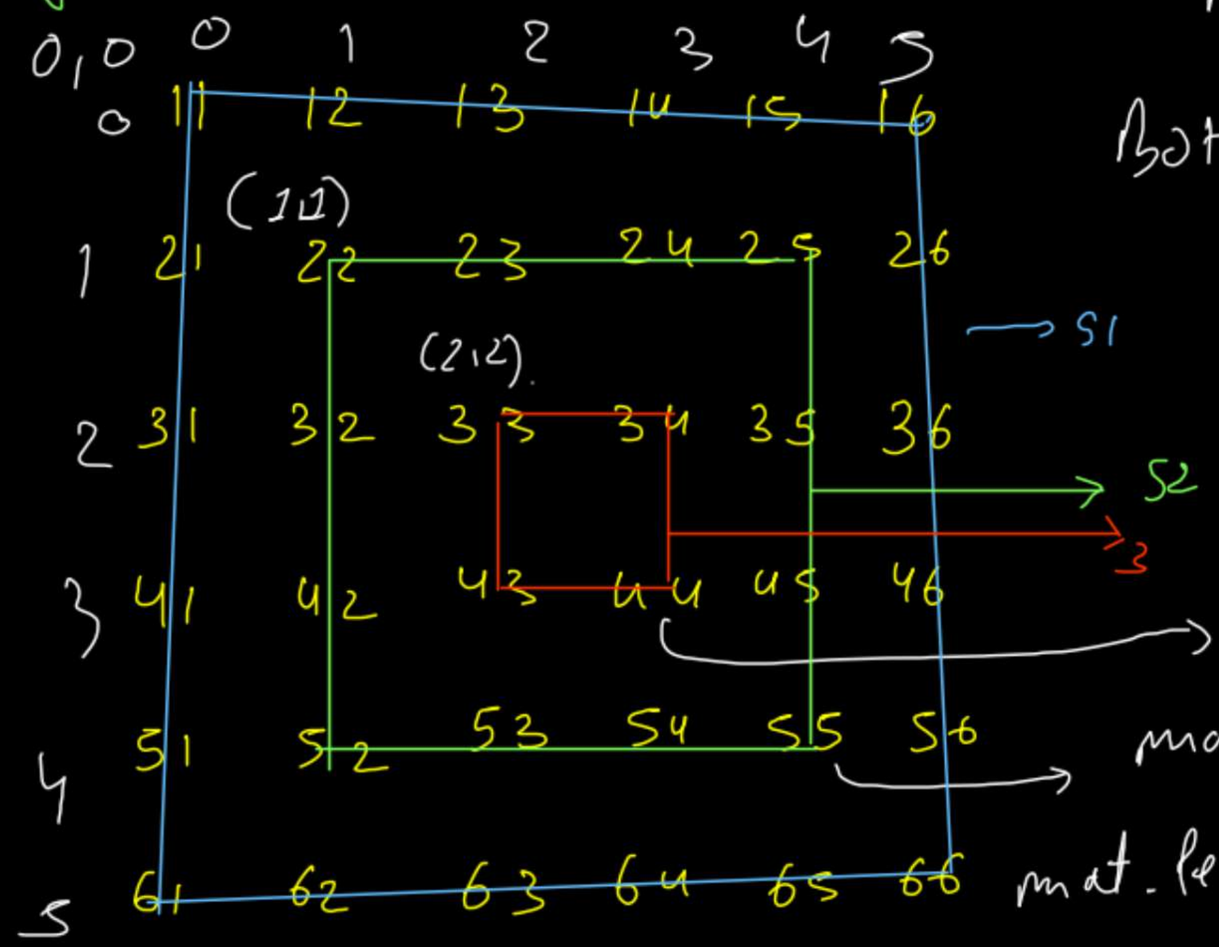
$\hookrightarrow$ max value in col 1

$a_{23} \rightarrow$ min value in row 2

$\hookrightarrow$ max value in col 3

H.W

Ring Rotate



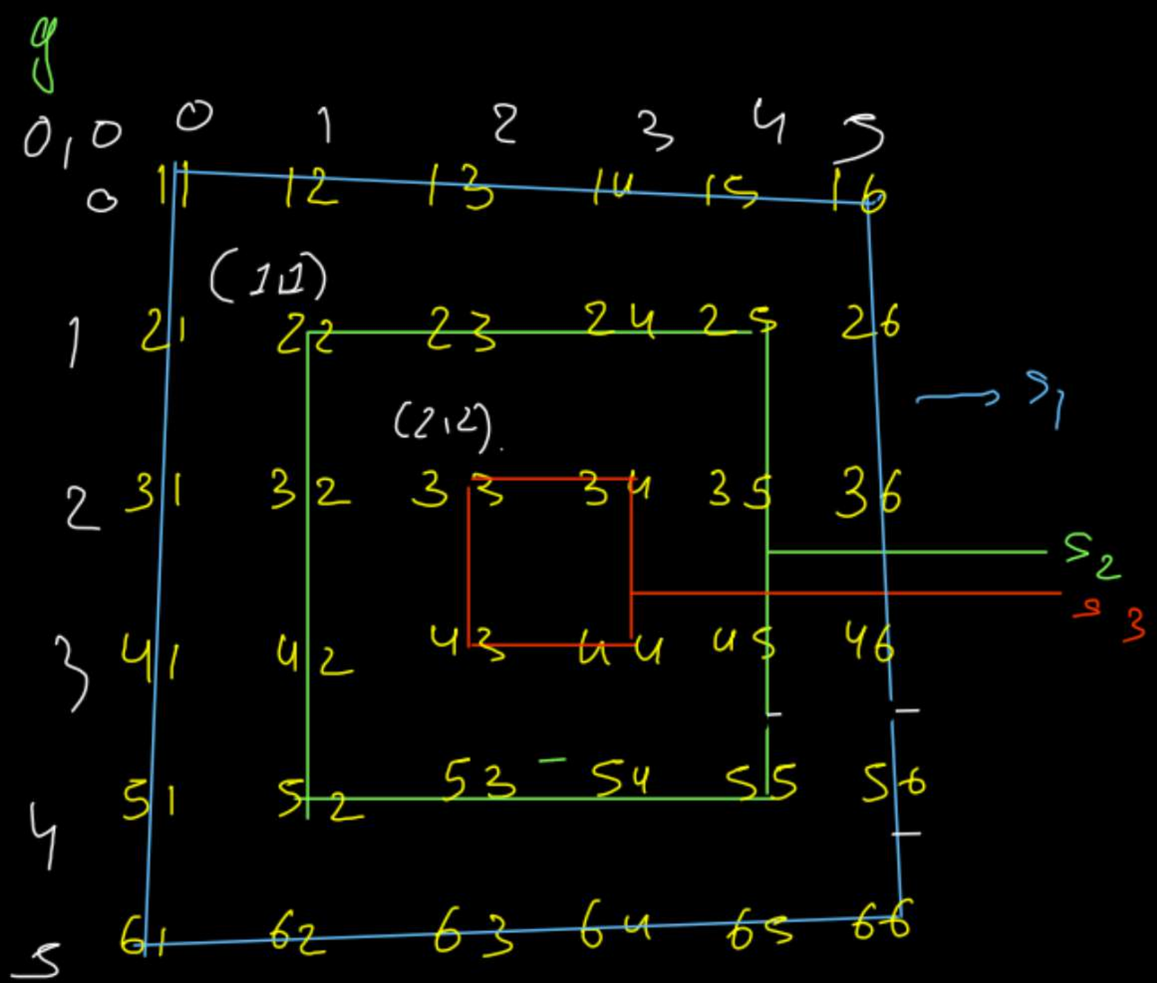
Top Left Corner = s-1, s-1

Bottom Right Corner = Rows - s, Cols - s

mat.length - 3, mat[0].length - 3

mat.length - 2, mat[0].length - 2

mat.length - 1, mat[0].length - 1



$$lw + bw + rw + tw - 4$$

$$2 \times lw + 2 \times bw - 4$$

$$2(lr - fr + 1)$$

$$+ 2(lc - fc + 1) - 4$$

$$2(lr - fr + 1)$$

$$+ 2(lc - fc + 1) - 4$$

$$\Rightarrow \quad 2lr - 2fr + \cancel{2} + 2lc - 2fc + \cancel{2} - \cancel{4}$$

$$= 2(lr - fr + lc - fc)$$

```java
public static void main(String[] args) throws Exception {
    // write your code here
    Scanner scn = new Scanner(System.in);
    int rows = scn.nextInt();
    int cols = scn.nextInt();

    int[][] arr = new int[rows][cols];

    for(int i=0;i<rows;i++) {
        for(int j=0;j<cols;j++) {
            arr[i][j] = scn.nextInt();
        }
    }

    int s = scn.nextInt();
    int r = scn.nextInt();

    shellRotate(arr,s,r);
    display(arr);
}
```

```java
public static void shellRotate(int[][] arr,int s, int r) {
    int[] oneD = fillOneDFromShell(arr,s);
    rotate(oneD,r);
    fillShellFromOneD(arr,s,oneD);
}

public static void rotate(int[] arr, int k) {
    int n = arr.length;

    k = k % n;
    if(k < 0) k = k + n;

    reverse(arr,n-k,n-1);
    reverse(arr,0,n-k-1);
    reverse(arr,0,n-1);
}

public static void reverse(int[] arr,int lo, int hi) {
    while(lo < hi) {
        int temp = arr[lo];
        arr[lo] = arr[hi];
        arr[hi] = temp;
        lo++;
        hi--;
    }
}

public static void display(int[][] arr){
    for(int i = 0; i < arr.length; i++){
        for(int j = 0; j < arr[0].length; j++){
            System.out.print(arr[i][j] + " ");
        }
        System.out.println();
    }
}

}
```

```java
public static int[] fillOneDFromShell(int[][] arr, int s) {
    int fr = s-1;
    int fc = s-1;
    int lr = arr.length -s;
    int lc = arr[0].length -s;

    int size = 2 * (lr - fr + lc - fc);
    int[] oneD = new int[size];
    int idx = 0;

    //first col
    for(int i=fr;i<=lr;i++) {
        oneD[idx] = arr[i][fc];
        idx++;
    }
    fc++;

    //last row
    for(int j=fc;j<=lc;j++) {
        oneD[idx] = arr[lr][j];
        idx++;
    }
    lr--;

    //last col
    for(int i=lr;i>=fr;i--) {
        oneD[idx] = arr[i][lc];
        idx++;
    }
    lc--;

    //first row
    for(int j=lc;j>=fc;j--) {
        oneD[idx] = arr[fr][j];
        idx++;
    }

    fr++;

    return oneD;
}
```

```java
public static void fillShellFromOneD(int[][] arr, int s, int[] oneD) {
    int fr = s-1;
    int fc = s-1;
    int lr = arr.length -s;
    int lc = arr[0].length -s;

    int idx = 0;

    //first col
    for(int i=fr;i<=lr;i++) {
        arr[i][fc] = oneD[idx];
        idx++;
    }
    fc++;

    //last row
    for(int j=fc;j<=lc;j++) {
        arr[lr][j] = oneD[idx];
        idx++;
    }
    lr--;

    //last col
    for(int i=lr;i>=fr;i--) {
        arr[i][lc] = oneD[idx];
        idx++;
    }
    lc--;

    //first row
    for(int j=lc;j>=fc;j--) {
        arr[fr][j] = oneD[idx];
        idx++;
    }

    fr++;
}
```

Lecture - 15

Strings in Java

Interning & its need

Implications of String Interning

- immutability
- == vs equals

String Builders & String Buffers

Strings in Java

String : A string is a Class in Java. So, just like any other class, it will have some data members & member functions. However, in simple terms, we can say that a String is an array of characters.

- ↳ lowercase ('a' - 'z')
- ↳ uppercase ('A' - 'Z')
- ↳ digits ('0' - '9')
- ↳ special symbols ('#', '*', etc)

↳ null character ('\0')

Program to Create and Use Strings in Java.

```java
public static void main(String[] args) {
    String str = "DSA"; 1
    String str1 = new String("Dev"); 2
    System.out.println(str);
    System.out.println(str1);
}
```

```java
public class Main {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        String str = scn.next(); // after the space, no input will be taken
        System.out.println(str);
    }
}
```

stdin

Hello, my name is Khan

```
Finished in 131 ms
Hello,
```

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

```
Finished in 130 ms
Hello, my name is Khan
```

stdin

```
Hello, my name is Khan
```

length() and charAt() Methods

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

    for(int i=0;i<str.length();i++) {
        char ch = str.charAt(i);
        System.out.println(ch);
    }
}
```

"Hello" → { 'H'(0), 'e'(1), 'l'(2), 'l'(3), 'o'(4) }

Each and every character can be represented by a numeric code.
This is called its ASCII value.

American Standard Code for Information Interchange.

** Java uses Unicode (code for many languages). Thus, the size of char datatype is 2bytes in Java.

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

https://unicode-table.com/en/ → for unicode Table

'0' – '9'
48 57

'A' – 'Z'
65 90

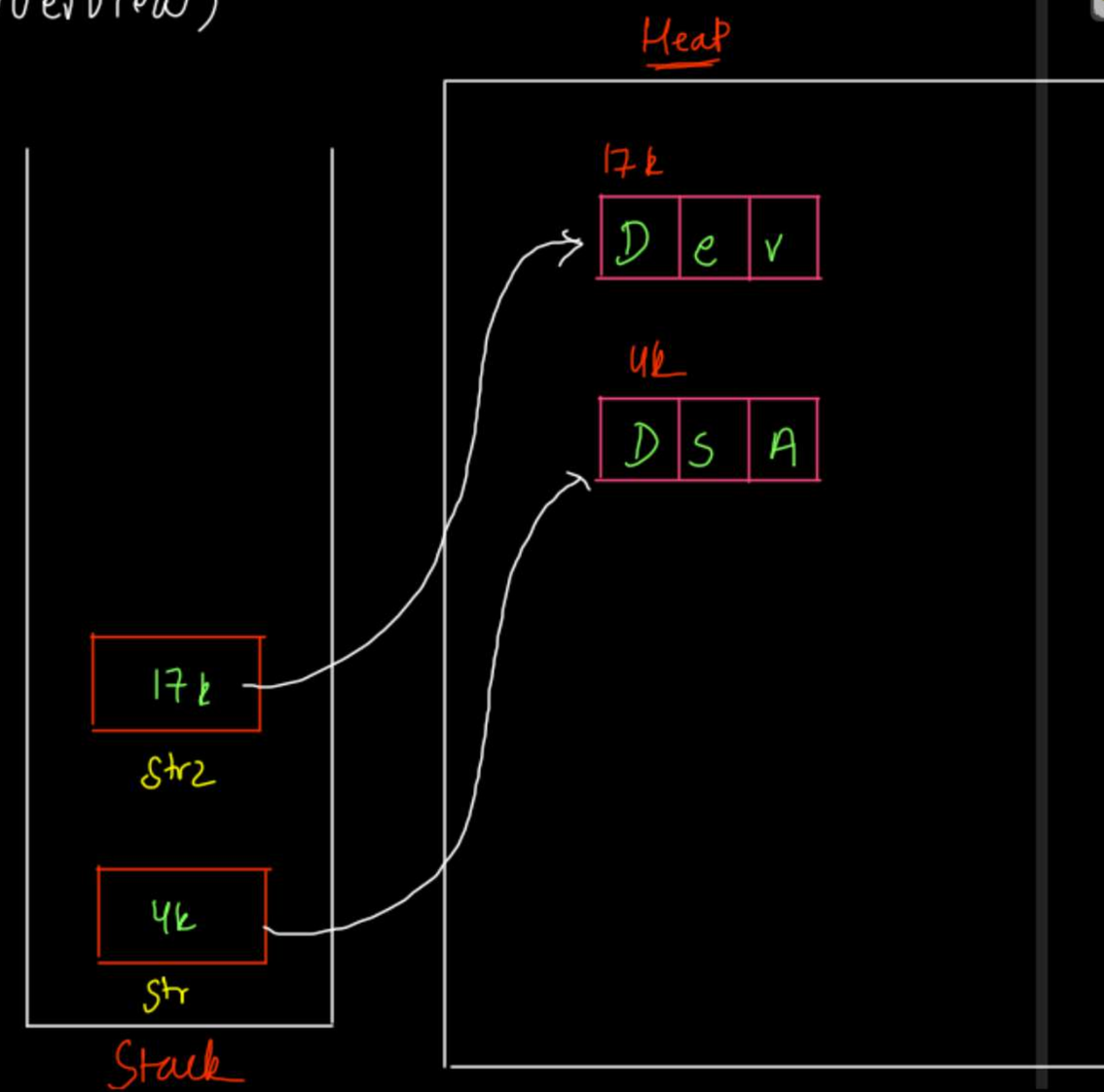
'a' – 'z'
97 122

+1 'a' – 'A' = 32

'A' – 'a' = -32

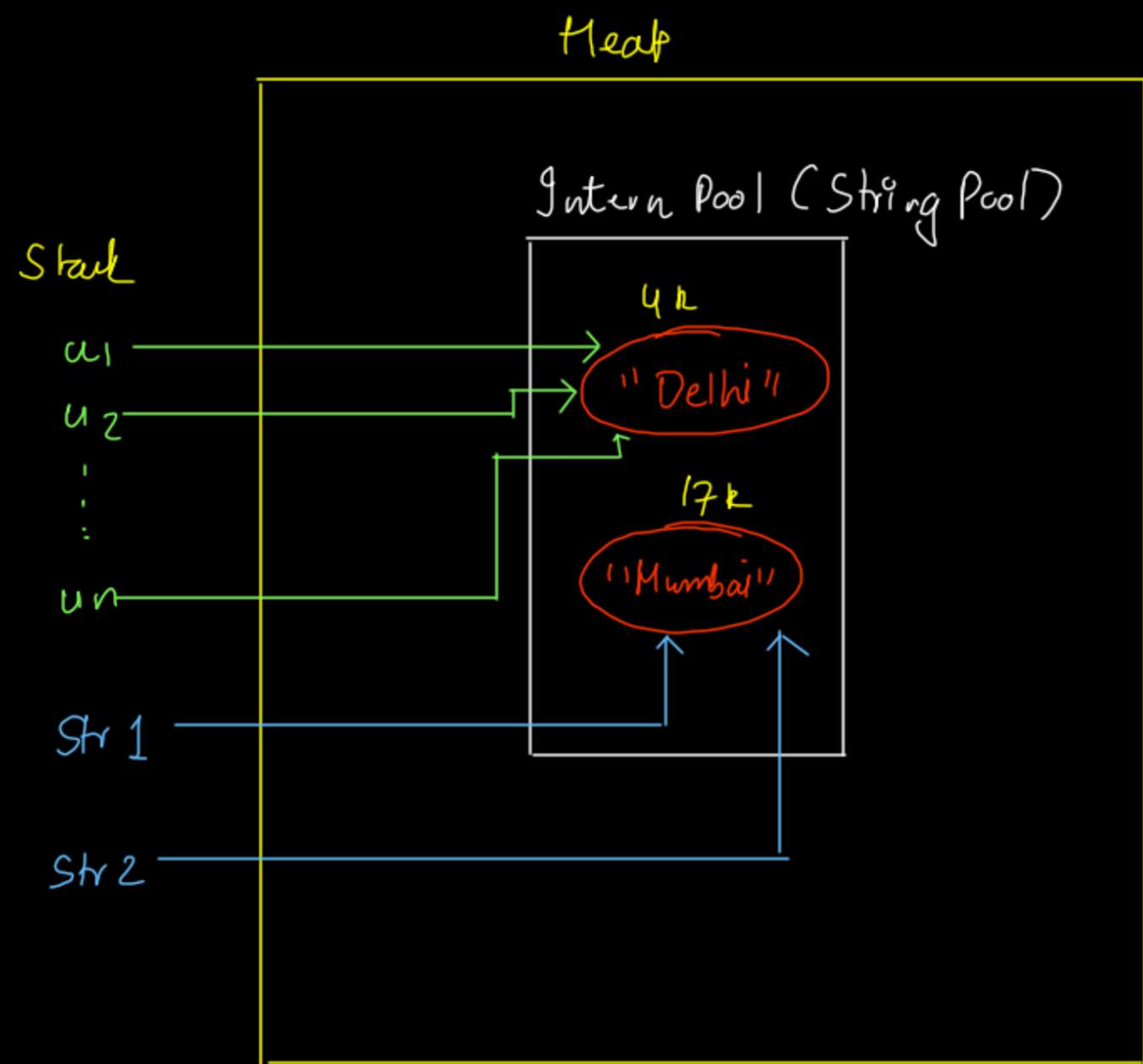
Strings Memory Mapping (Only Overview)

```
String str = "DSA";

String str2 = new String("Dev");
```



Strings Interning

```
String u1 = "Delhi";
String u2 = "Delhi";
.
.
.
String un = "Delhi";

String str1 = "Mumbai";
String str2 = "Mumbai";
```



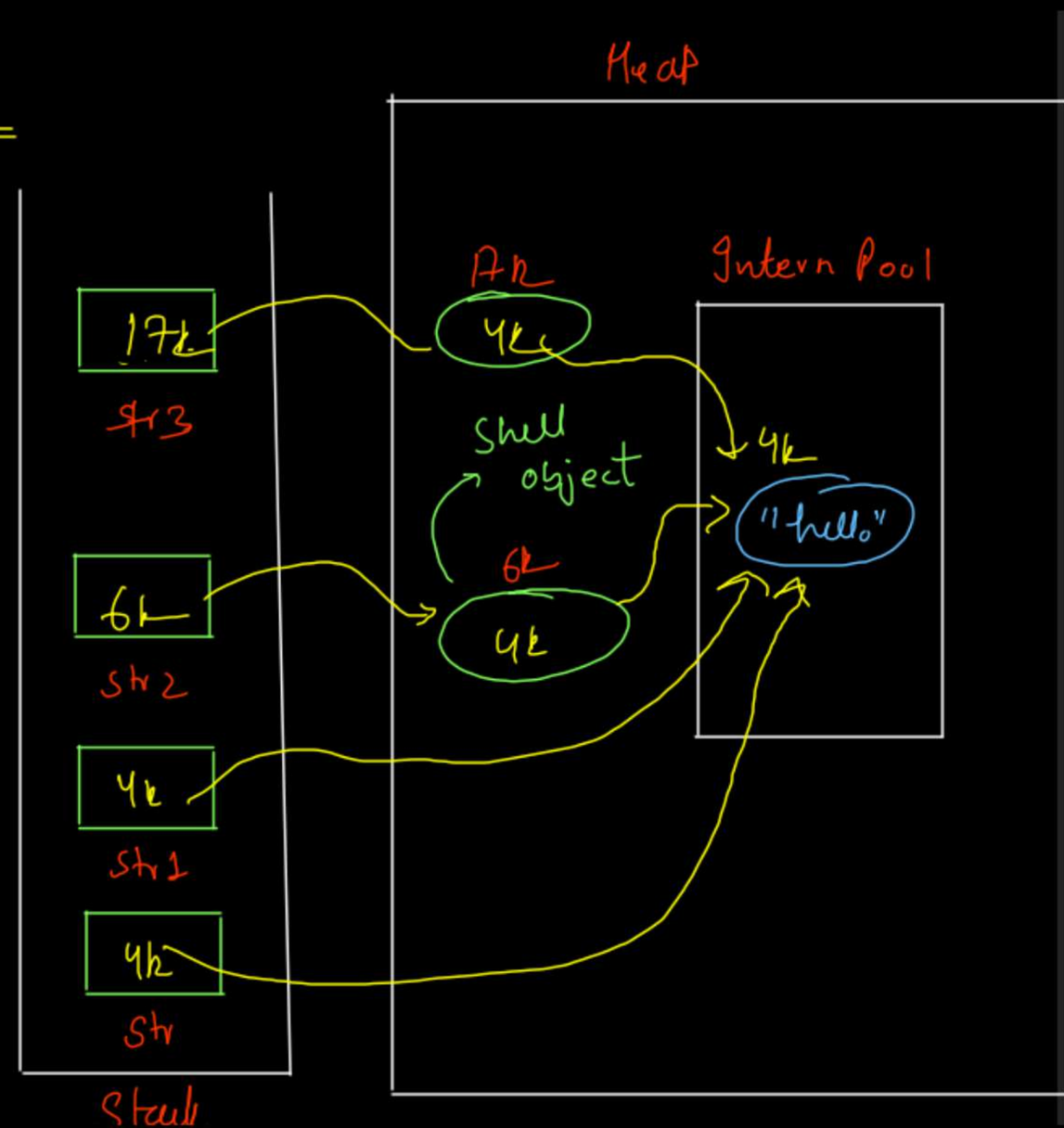
In Java, string literals are created inside a special memory section inside the heap memory called the intern pool or the string pool.

Interning is the concept of using the same string literal i.e. the same string literal is referenced by many references to save memory like u1, u2,... un in the previous example were all referencing to the same string literal "Delhi".

Interning in Java has 2 main implications
1. Java Strings are immutable
2. == vs equals in Strings

String Complete Memory Mapping

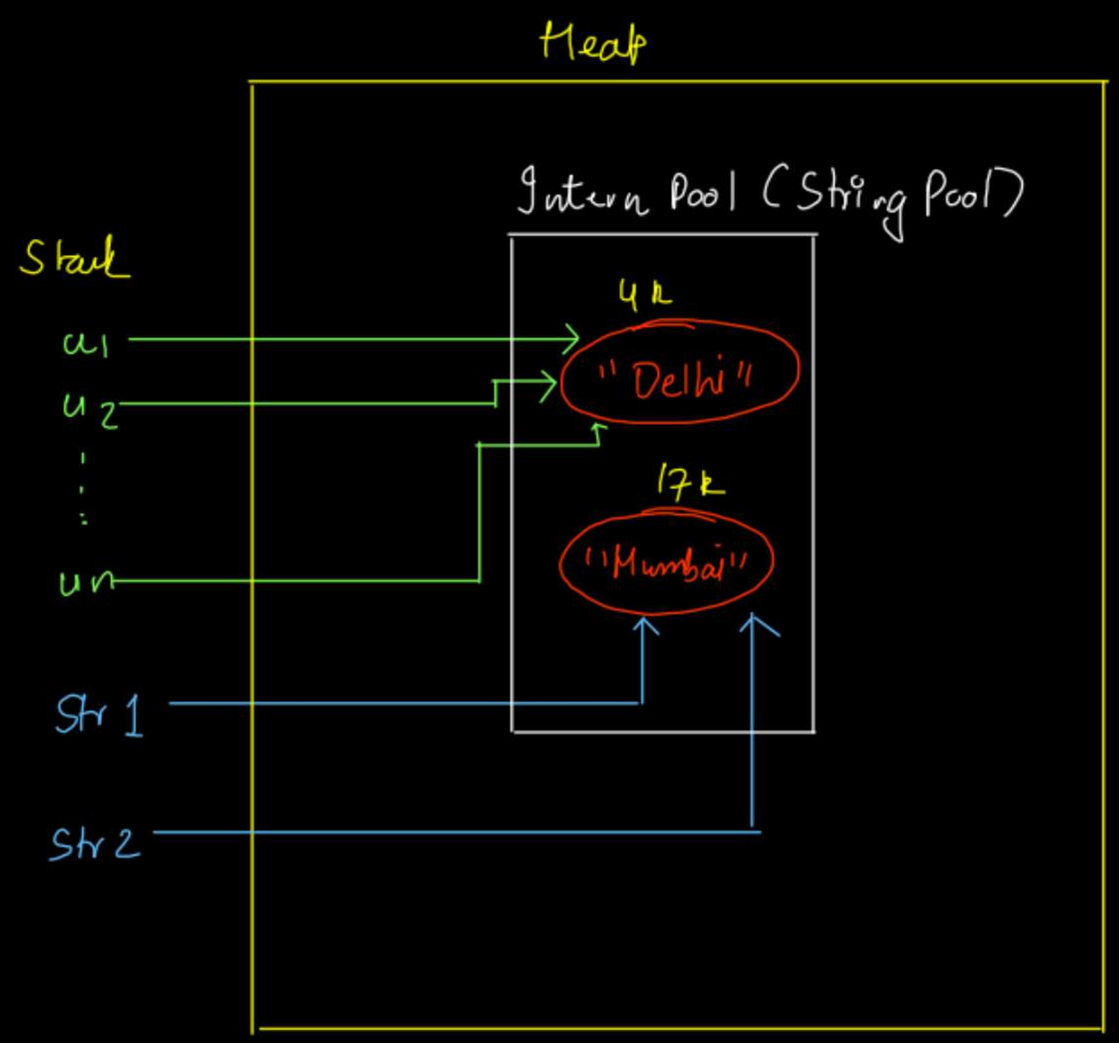
```
String str = "hello";
String str1 = "hello";
String str2 = new String("hello");
String str3 = new String("hello");
```



When we directly assign the String literal to a reference, if the literal already exists in the intern pool, the reference points to the already existing literal due to interning. Otherwise, a new literal is created in the intern pool and the reference stores it's address.

Instead of directly assigning a String literal, if the new keyword is used, a shell object is created which further stores the address of the String literal and our reference stores the address of this shell object. This shell object is created inside heap but not inside the intern pool. Whenever the new keyword is used, a new shell object will be created, however, the same string literal will be used if it exists due to interning.

Implications of Interning – Strings are immutable.

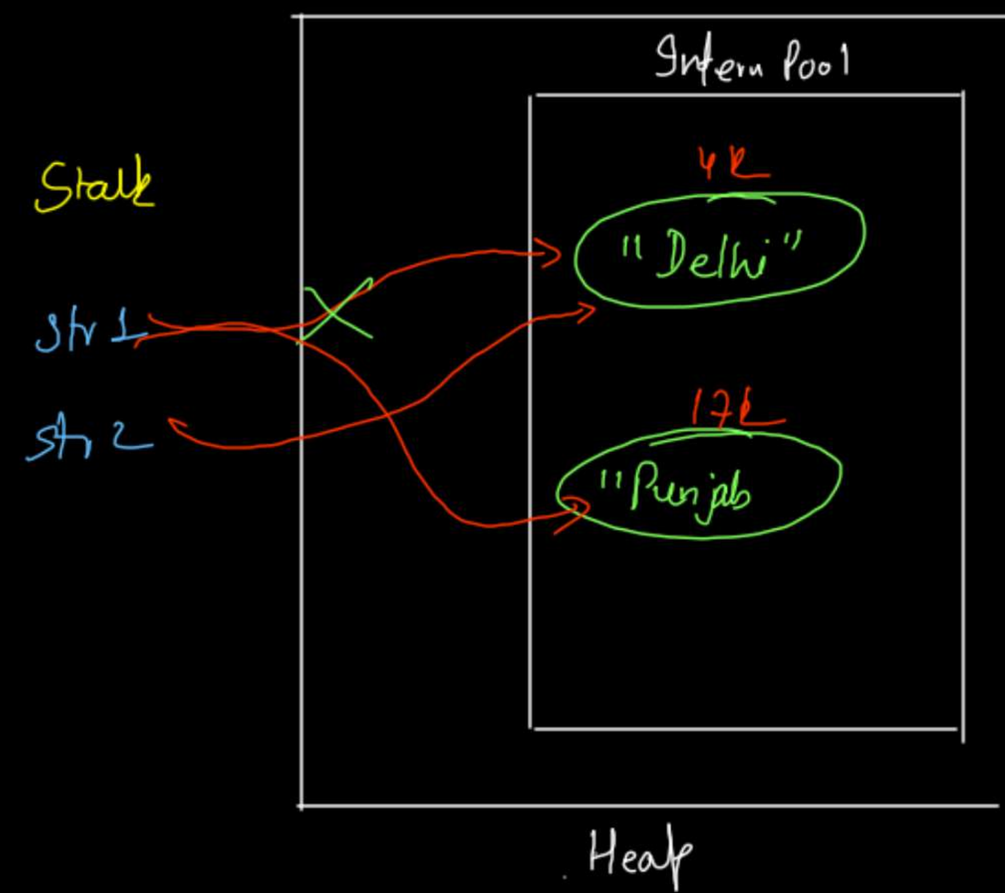


Need of immutability

u_1 = "Punjab";

** References are mutable, instances are not.

```java
public class Main {

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

        System.out.println(str1);
        System.out.println(str2);

        str1 = "Punjab";

        System.out.println(str1);
        System.out.println(str2);

    }
```




```java
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        String str = "h";
        str = str + 'e';
        str  = str + 'l';
        str  = str + 'l';
        str = str + "o";
        System.out.println(str);
    }
}
```

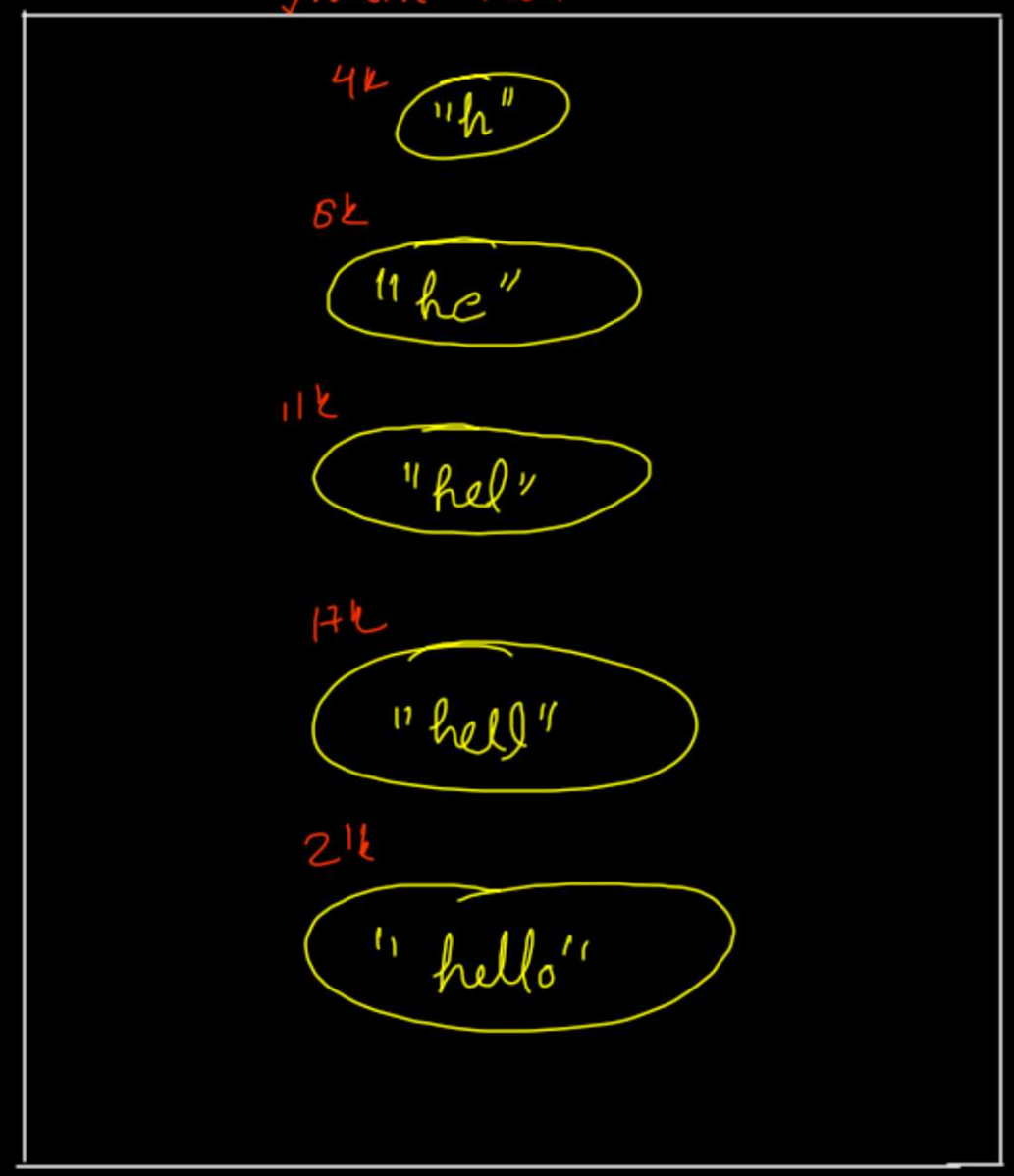
```
str = "h";
str = str + "e";
str = str + "l";
str = str + "l";
str = str + "o";
```

$O(N)$

$O(N^2)$

str = ~~4k~~ ~~8k~~ ~~11k~~ ~~17k~~ 21k

Intern Pool



Hello → 5

1. H
2. e → 1(H)
3. l → 2(He)
4. l → 3(Hel)
5. o → 4(Hell)

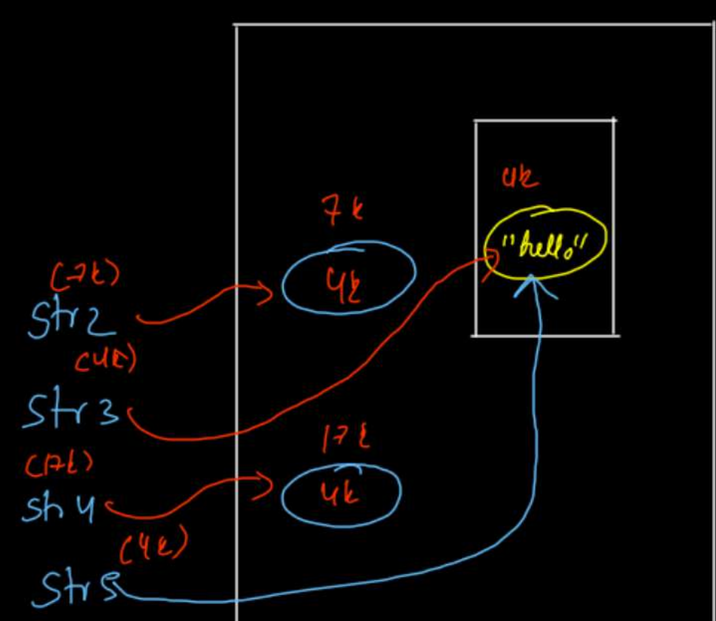
$$\frac{N(N-1)}{2} = O(N^2)$$

System.gc

Implications of Interning — equals vs ==

LHS == RHS

```java
import java.util.*;

public class Main {
    public static void main(String[] args) {
        // String str1 = "Guneet";
        String str2 = new String("hello");
        String str3 = "hello";
        String str4 = new String("hello");
        String str5 = "hello";

        System.out.println(str2 == str3); → false
        System.out.println(str3 == str5); → true
        System.out.println(str2 == str4); → false
        System.out.println(str2.equals(str3)); → true
        System.out.println(str2.equals(str4)); → true
        System.out.println(str2.equals(str5)); → true
    }
}
```



StringBuilders & String Buffers

You can say that they are just like Strings in C++. Here, references and instances, both are mutable.

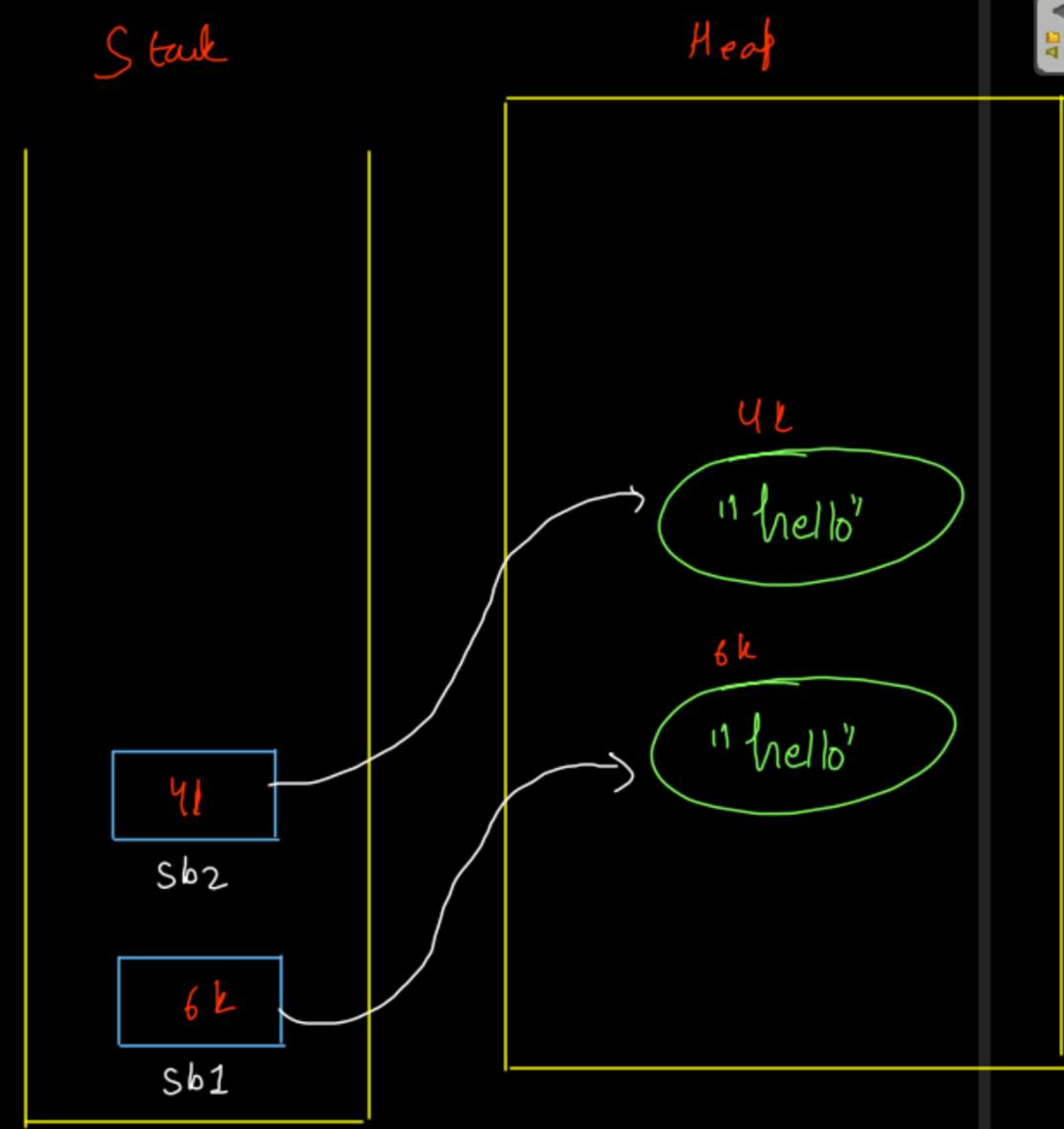
It is also possible to update (change) a character at a particular index.

There is no interning in StringBuilders and String Buffers.

** Difference b/w a StringBuilder and StringBuffer is that the StringBuffer is a thread-safe class whereas StringBuilder is not.

String Builder (Memory Mapping)

```
StringBuilder sb1 = new StringBuilder("hello");
StringBuilder sb2 = new StringBuilder("hello");
```



StringBuilders (Usage & functions)

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    StringBuilder sb = new StringBuilder("Hello");
    System.out.println(sb);

    sb.setCharAt(0,'h');
    System.out.println(sb);
    sb.append('o');
    System.out.println(sb);

    sb.append(",there!");
    System.out.println(sb);
    sb.append(1);
    System.out.println(sb);

    String s = sb.toString();
    System.out.println(s);
}
```

Lecture-16

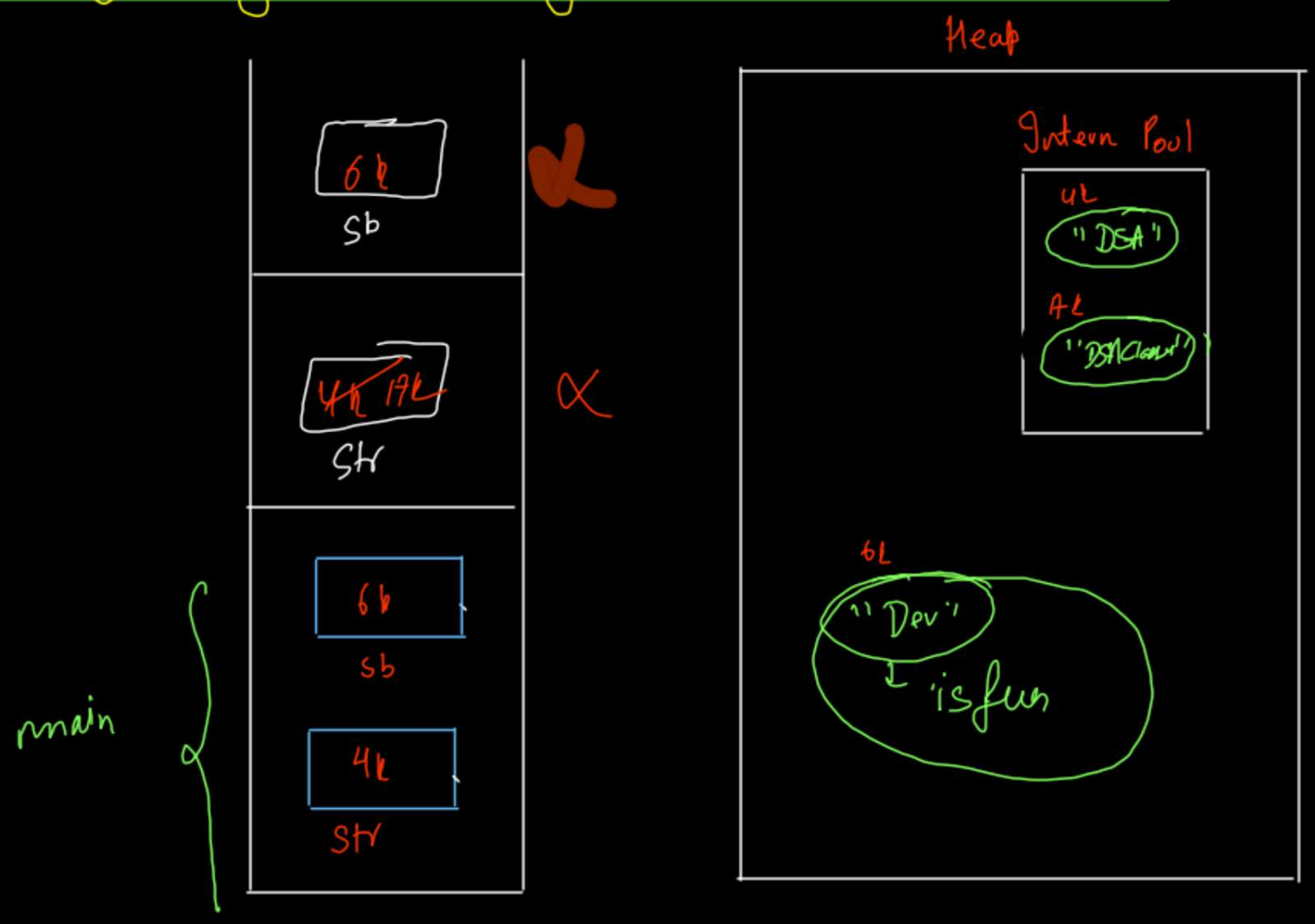
- → Passing String & StringBuilder into functions
- → Substring Method
- → Print All Palindromic Substrings
- → Reverse String
- → Change the String (Toggle Case)
- → String Compression
- → Reverse Words.

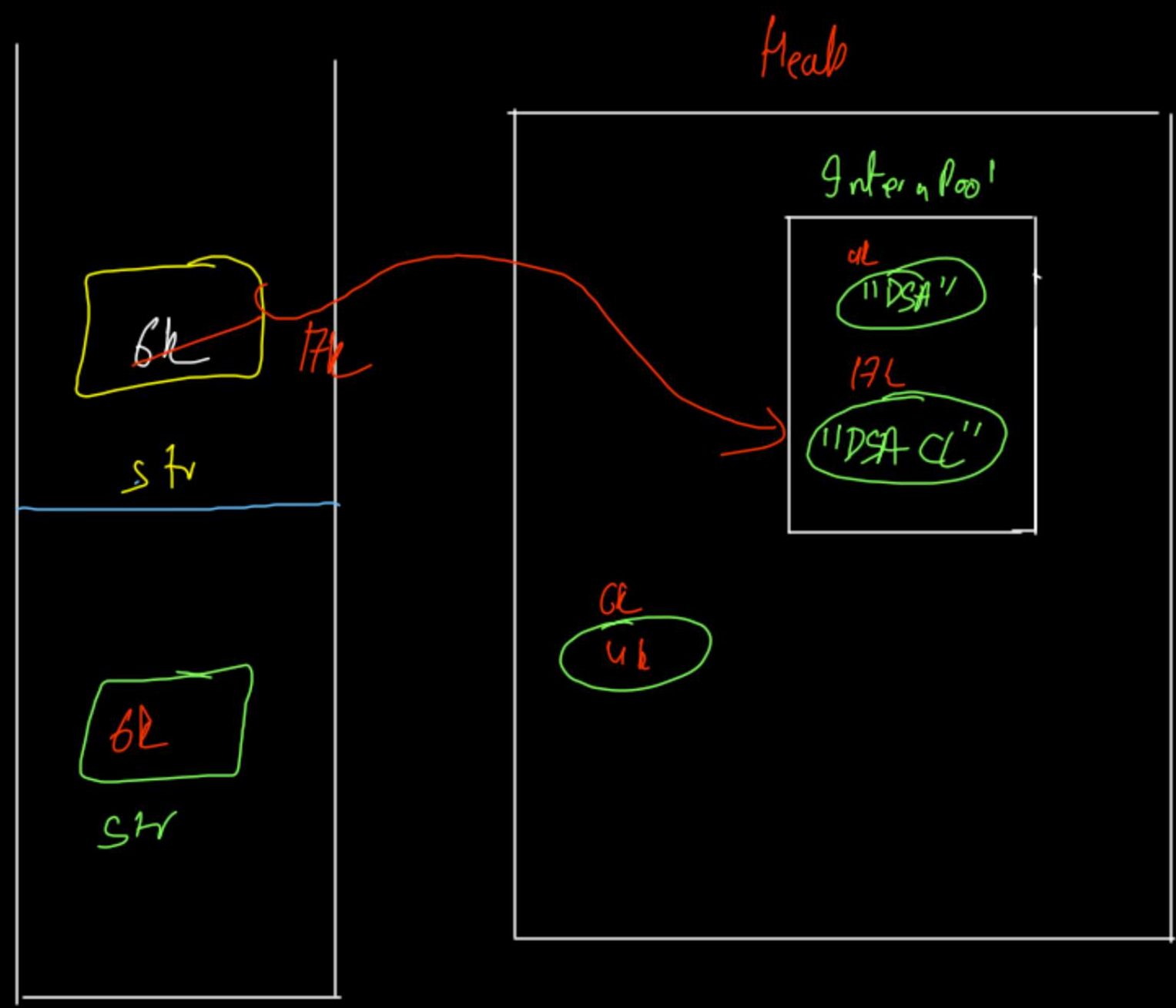
Passing String and StringBuilders into functions.

```java
import java.util.*;

public class Main {

    public static void modifyString(String str) {
        str = str + " Classes";
    }

    public static void modifyStringBuilder(StringBuilder sb) {
        sb.append(" is fun");
    }

    public static void main(String[] args) {
        String str = "DSA";
        StringBuilder sb = new StringBuilder("Dev");
        modifyString(str);
        modifyStringBuilder(sb);

        System.out.println(str);
        System.out.println(sb);
    }
}
```

Output ?

Passing String and StringBuilders into functions. (Memory Mapping)




Heap
InternPool
"DSA"
"DSA CL"
str
str

Subarrays

0 1 2

{1, 2, 3}

1

1, 2

1, 2, 3

2

2, 3

3

```
for(i=0; i<n ; i++)
{ for(j=i ; j<n ; j++)
  {
    for (k=i; k<=j ; k++)
    {
      }
```

String → substring() method

str = "abcdef" (0 1 2 3 4 5)

str.substring(2); → cdef

str.substring(start, end); → Returns a string (substring) which has the characters from our string str starting from index start till index end-1.

(1, 2) → b

(1, 4) → bcd

(0, 5) → abcde

(0, 6) → abcdef

Print all Substrings

```java
public static void main(String[] args) {
    String str = "abc";

    for(int i=0;i<str.length();i++) {
        for(int j=i;j<str.length();j++) {
            String subStr = str.substring(i,j + 1);
            System.out.println(subStr);
        }
    }
}
```

(6 1) → empty / blank string string

→ str.length() = 0

Print All Palindromic Substrings.

Print All Palindromic Substrings

Easy

Prev Next

1. You are given a string.
2. You have to print all palindromic substrings of the given string.

Input Format

A String

Output Format

All palindromic substrings(one in a line).
First, all palindromic substrings starting from first character of string will be printed, then from second character and so on.

"naan"

hi

~~lo~~ ~~lo~~ lo ~~hi~~ ~~hi~~

"naman"

0 1 2 3 4

while(lo< hi)

"~~lo~~ lo hi ~~hi~~"

nabn

hi lo

~~lo~~ ~~lo~~ ~~hi~~ ~~hi~~

"naan"

0 1 2 3

```
public static boolean isPalindrome(String str) {
    int lo = 0;
    int hi = str.length()-1;

    while(lo < hi) {
        if(str.charAt(lo) != str.charAt(hi))
        return false;
        lo++;
        hi--;
    }

    return true;
}

public static void solution(String str){
    //write your code here

    for(int i=0;i<str.length();i++) {
        for(int j=i;j<str.length();j++) {
            String subStr = str.substring(i,j+1);
            if(isPalindrome(subStr)) {
                System.out.println(subStr);
            }
        }
    }

}
```

Reverse String

= ["h", "e", "l", "l", "o"]

olleh

```java
class Solution {
    public void reverseString(char[] s) {
        int lo = 0;
        int hi = s.length -1;

        while(lo < hi) {
            char temp = s[lo];
            s[lo] = s[hi];
            s[hi] = temp;
            lo++;
            hi--;
        }
    }
}
```

Change the String

Change the string

Basic Accuracy: **61.52%** Submissions: **11111** Points: **1**

Given a string S, the task is to change the complete string to Uppercase or Lowercase depending upon the case for the first character.

Example 1:

Input:
S = "abCD"
Output: abcd
Explanation: The first letter (a) is lowercase. Hence, the complete string is made lowercase.

"abCD"

ch = 'a'

a = A+32

A = a-32

a - A = 32

b - B = 32

c - C = 32

long

int

```java
class Solution{
    String modify(String s){
        if(s.length() == 0 || s == null) return s;

        boolean isLowerCase = false;
        if(s.charAt(0) >= 'a' && s.charAt(0) <= 'z') isLowerCase = true;

        StringBuilder sb = new StringBuilder("");

        for(int i=0;i<s.length();i++) {
            char ch = s.charAt(i);
            if(ch >= 'a' && ch <= 'z' && isLowerCase == false) {
                ch = (char)(ch - 32);
            } else if(ch >= 'A' && ch <= 'Z' && isLowerCase == true) {
                ch = (char)(ch + 32);
            }
            sb.append(ch);
        }

        return sb.toString();
    }
```

0 1 2 3 4 5
a A B c D E

LC = true

ch = 'A'
a

aabc

Reverse words in a given string

Easy Accuracy: **50.0%** Submissions: **100k+** Points: **2**

This problem is part of GFG SDE Sheet. Click here to view more.

Given a String S, reverse the string without reversing its individual words. Words are separated by dots.

Example 1:

Input:
S = i.like.this.program.very.much
Output: much.very.program.this.like.i
Explanation: After reversing the whole string(not individual words), the input string becomes much.very.program.this.like.i

j = -1

i = -2

11

i. like. this. program

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18

program. this. like. i

```
String reverseWords(String S)
{
    // code here
    StringBuilder sb = new StringBuilder("");
    int i = S.length()-1;

    while(i >= 0) {
        int j = i-1;

        while(j >= 0 && S.charAt(j) != '.') {
            j--;
        }

        for(int x=j+1;x<=i;x++) {
            sb.append(S.charAt(x));
        }

        i = j-1;
        if(i >= 0) sb.append(".");
    }

    return sb.toString();

}
```

Lecture: 17

- → String Compression
- → Reverse String -2
- → Valid Palindrome -1
- → Valid Palindrome -2

443. String Compression

Medium 2398 4533 Add to List Share

Given an array of characters `chars`, compress it using the following algorithm:

Begin with an empty string `s`. For each group of **consecutive repeating characters** in `chars`:

- If the group's length is `1`, append the character to `s`.
- Otherwise, append the character followed by the group's length.

The compressed string `s` **should not be returned separately**, but instead, be stored **in the input character array** `chars`. Note that group lengths that are `10` or longer will be split into multiple characters in `chars`.

After you are done **modifying the input array,** return *the new length of the array*.

You must write an algorithm that uses only constant extra space.

j i i j

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

"aaaaaaaaaaaabbbbbbbbbcccccccccab"

j j j

```java
class Solution {
    public int compress(char[] chars) {
        int idx = 0;
        int i = 0;

        while(i < chars.length) {
            int j = i;

            while(j < chars.length && chars[j] == chars[i]) {
                j++;
            }

            chars[idx] = chars[i];
            idx++;

            String count = "" + (j-i);

            if(j-i > 1) {
               for(int x=0;x<count.length();x++) {
                   char ch = count.charAt(x);
                   chars[idx] = ch;
                   idx++;
               }
            }

            i = j;
        }

        return idx;
    }
}
```

541. Reverse String II

Easy 1222 2729 Add to List Share

Given a string `s` and an integer `k`, reverse the first `k` characters for every `2k` characters counting from the start of the string.

If there are fewer than `k` characters left, reverse all of them. If there are less than `2k` but greater than or equal to `k` characters, then reverse the first `k` characters and leave the other as original.

Example 1:

```
Input: s = "abcdefg", k = 2
Output: "bacdfeg"
```

Example 2:

```
Input: s = "abcd", k = 2
Output: "bacd"
```

i i

0 1 2 3 4 5 6 7 8 9 10

"abcdefghijk"

d c b a

lo hi

k = 4

i += 2 * k

2k

k k

k k

for current k elements that we have reversed

for next k elements that we don't have to reverse

lo = i

hi = min(length - 1, i + k - 1)

while(lo < hi) {

// swapping

}

```java
class Solution {
    public String reverseStr(String s, int k) {
        StringBuilder sb = new StringBuilder(s);

        for(int i=0;i<s.length(); i+= 2*k) {
            int lo = i;
            int hi = (int)Math.min(i+k,s.length()) - 1;

            while(lo < hi) {
                char temp = sb.charAt(lo);
                sb.setCharAt(lo,sb.charAt(hi));
                sb.setCharAt(hi,temp);
                lo++;
                hi--;
            }
        }

        return sb.toString();
    }
}
```

125. Valid Palindrome

Easy 4877 5989 Add to List Share

A phrase is a **palindrome** if, after converting all uppercase letters into lowercase letters and removing all non-alphanumeric characters, it reads the same forward and backward. Alphanumeric characters include letters and numbers.

Given a string `s`, return `true` *if it is a* ***palindrome****, or* `false` *otherwise.*

Example 1:

```
Input: s = "A man, a plan, a canal: Panama"
Output: true
Explanation: "amanaplanacanalpanama" is a palindrome.
```

```java
public boolean isLowerCase(char ch) {
    if(ch >='a' && ch <= 'z') return true;
    return false;
}

public boolean isUpperCase(char ch) {
    if(ch >='A' && ch <= 'Z') return true;
    return false;
}

public boolean isDigit(char ch) {
    if(ch >= '0' && ch <= '9') return true;
    return false;
}
```

```java
public String convert(String str) {

    StringBuilder sb = new StringBuilder("");

    for(int i=0;i<str.length();i++) {
        char ch = str.charAt(i);
        if(isLowerCase(ch) || isUpperCase(ch) || isDigit(ch)) {
            if(isUpperCase(ch)) {
                ch = (char)(ch + 32);
            }
            sb.append(ch);
        }
    }

    return sb.toString();
}
```

```java
public boolean isPalindrome(String s) {
    String str = convert(s);

    int lo = 0;
    int hi = str.length()-1;

    while(lo < hi) {
        if(str.charAt(lo) != str.charAt(hi)) return false;
        lo++;
        hi--;
    }

    return true;
}
```

680. Valid Palindrome II

Easy 6305 328 Add to List Share

Given a string `s`, return `true` *if the* `s` *can be palindrome after deleting* ***at most one*** *character from it.*

Example 1:

```
Input: s = "aba"
Output: true
```

Example 2:

```
Input: s = "abca"
Output: true
Explanation: You could delete the character 'c'.
```

"aba"

"abca"

lo lo lo lo lo lo hi hi hi hi hi hi

0 1 2 3 4 5 6 7 8 9 10 11 12 13

"abcdefghkedcba"

1st part 2nd part 3rd part

If string is a Palindrome,

1st part = 3rd part &

2nd part must be a palindrome.

```java
public boolean isPalindrome(String str) {

    int lo = 0;
    int hi = str.length()-1;

    while(lo < hi) {
        if(str.charAt(lo) != str.charAt(hi)) return false;
        lo++;
        hi--;
    }

    return true;
}
```

```java
  public boolean validPalindrome(String s) {
      int lo = 0;
      int hi = s.length()-1;

      while(lo < hi) {
          if(s.charAt(lo) == s.charAt(hi)) {
              lo++;
              hi--;
          } else {
              boolean ans1 = isPalindrome(s.substring(lo,hi));
              boolean ans2 = isPalindrome(s.substring(lo+1,hi+1));
              return ans1 || ans2;
          }
      }

      return true;
  }
}
```

Lecture - 18

→ Reverse Vowels

→ Reverse Words - 3

→ Reverse Words - 2

→ Arraylist (Syntax)

→ Wrapper Class → Autoboxing, Unboxing

Reverse Vowels

345. Reverse Vowels of a String

Easy 1910 2000 Add to List Share

Given a string `s`, reverse only all the vowels in the string and return it.

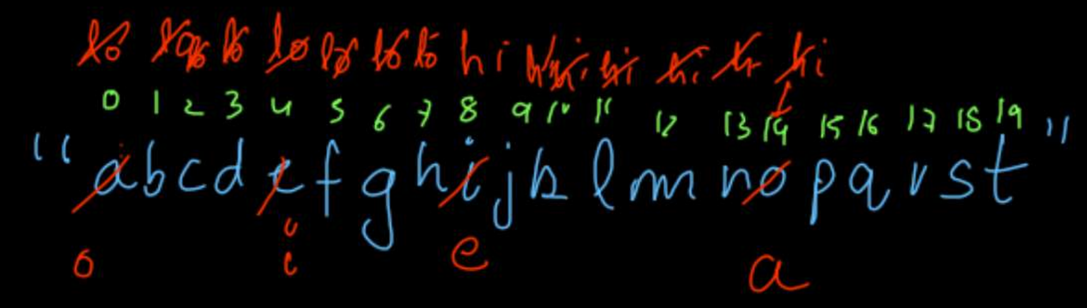
The vowels are `'a'`, `'e'`, `'i'`, `'o'`, and `'u'`, and they can appear in both cases.

Example 1:

```
Input: s = "hello"
Output: "holle"
```

Example 2:

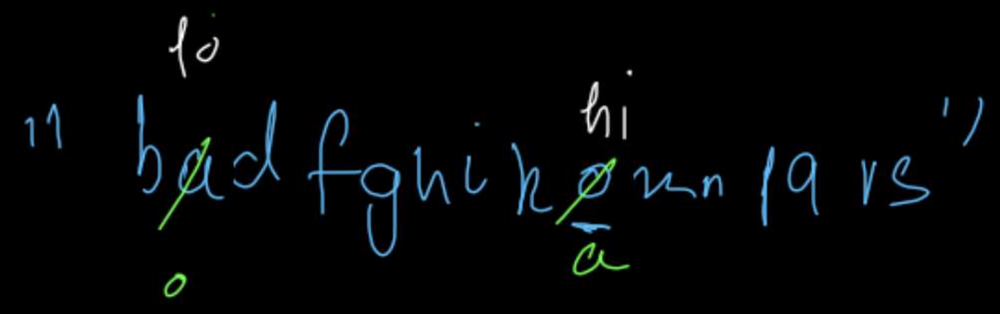
```
Input: s = "leetcode"
Output: "leotcede"
```



a, e, i, o & u

lo = 0

hi = sb.length() -1




```java
public boolean isVowel(char ch) {
    if(ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') {
        return true;
    } else if(ch == 'A' || ch == 'E' || ch == 'I' || ch == 'O' || ch == 'U') {
        return true;
    }

    return false;
}
```

```java
public String reverseVowels(String s) {
    int lo = 0;
    int hi = s.length() -1;

    StringBuilder sb = new StringBuilder(s);

    while(lo < s.length() && isVowel(s.charAt(lo)) == false) {
        lo++;
    }

    while(hi >= 0 && isVowel(s.charAt(hi)) == false) {
        hi--;
    }

    while(lo < hi) {
        char temp = sb.charAt(lo);
        sb.setCharAt(lo,sb.charAt(hi));
        sb.setCharAt(hi,temp);

        lo++;
        hi--;

          while(lo < s.length() && isVowel(s.charAt(lo)) == false) {
            lo++;
          }

          while(hi >= 0 && isVowel(s.charAt(hi)) == false) {
              hi--;
          }
    }

    return sb.toString();
}
```

557. Reverse Words in a String III

Easy 4103 209 Add to List Share

Given a string `s`, reverse the order of characters in each word within a sentence while still preserving whitespace and initial word order.

Example 1:

```
Input: s = "Let's take LeetCode contest"
Output: "s'teL ekat edoCteeL tsetnoc"
```

Example 2:

```
Input: s = "God Ding"
Output: "doG gniD"
```

"Hello world"

"olleH_ dlrow"

"Hello world"

"olleH_ dlrow"

$i = j + 1$

$i < s.length()$

```java
class Solution {
    public String reverseWords(String s) {
        int i = 0;
        StringBuilder sb = new StringBuilder("");

        while(i < s.length()) {
            int j = i + 1;

            while(j < s.length() && s.charAt(j) != ' ') {
                j++;
            }

            for(int x=j-1;x>=i;x--) {
                sb.append(s.charAt(x));
            }

            i = j + 1;
            if(i < s.length()) sb.append(' ');
        }

        return sb.toString();
    }
}
```

151. Reverse Words in a String

Medium 4002 4140 Add to List Share

Given an input string `s`, reverse the order of the **words**.

A **word** is defined as a sequence of non-space characters. The **words** in `s` will be separated by at least one space.

Return *a string of the words in reverse order concatenated by a single space.*

Note that `s` may contain leading or trailing spaces or multiple spaces between two words. The returned string should only have a single space separating the words. Do not include any extra spaces.

i

j

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

"the _ _ sky _ _ is _ _ _ blue _ _"

i = j;

"blue _ is _ sky _ the"

j i

blue _ _

j i

_ _ the

the

```java
class Solution {
    public String reverseWords(String s) {
        int i = s.length()-1;
        StringBuilder sb = new StringBuilder("");

        while(i >= 0 && s.charAt(i) == ' ') {
            i--;
        }

        while(i >= 0) {
            int j = i-1;

            while(j >= 0 && s.charAt(j) != ' ') {
                j--;
            }

            for(int x=j+1;x<=i;x++) {
                sb.append(s.charAt(x));
            }

            while(j >= 0 && s.charAt(j) == ' ') {
                j--;
            }

            i = j;
            if(i >= 0) sb.append(' ');
        }

        return sb.toString();
    }
}
```

ArrayList

Arrays are static.

int [] arr =

0	1	2	3	4
1	2	3	4	5

6

Array Lists are dynamic arrays. (Size can be modified (automatically))

// Declare & Initialization

```java
import java.util.*;

public class Main {

    public static void main(String[] args) {

        //declaration
        ArrayList<Integer> arr;
        //initialization
        arr = new ArrayList<>();

        System.out.println(arr);


    }
}
```

This does not print any Hashcode. You can directly print AL in Java.

```
Finished in 87 ms
[]
```

//Add (Insert at End (Append))

```java
public static void main(String[] args) {

    //declaration
    ArrayList<Integer> arr;
    //initialization
    arr = new ArrayList<>();

    arr.add(10);
    System.out.println(arr);

    arr.add(20);
    System.out.println(arr);

    arr.add(30);
    System.out.println(arr);


}
```

```
Finished in 87 ms
[10]
[10, 20]
[10, 20, 30]
```

// Size → Returns the size of AL (current no of ele)

```java
public static void main(String[] args) {

    //declaration
    ArrayList<Integer> arr;
    //initialization
    arr = new ArrayList<>();

    arr.add(10);
    System.out.println(arr.size());
    System.out.println(arr);

    arr.add(20);
    System.out.println(arr.size());
    System.out.println(arr);

    arr.add(30);
    System.out.println(arr.size());
    System.out.println(arr);

}
```

```
Finished in 93 ms
1
[10]
2
[10, 20]
3
[10, 20, 30]
```

Traversal & get() method.

```java
public class Main {

    public static void main(String[] args) {

        //declaration
        ArrayList<Integer> arr;
        //initialization
        arr = new ArrayList<>();

        arr.add(10);
        arr.add(20);
        arr.add(30);

        for(int i=0;i<arr.size();i++) {
            System.out.println(arr.get(i));
        }

    }
}
```

Returns the value at a particular index.

for Each

```java
public class Main {

    public static void main(String[] args) {

        //declaration
        ArrayList<Integer> arr;
        //initialization
        arr = new ArrayList<>();

        arr.add(10);
        arr.add(20);
        arr.add(30);

//          for(int i=0;i<arr.size();i++) {
//              System.out.println(arr.get(i));
//          }

        for(int ele : arr) {
            System.out.println(ele);
        }
    }
```

0 1 2 3 4

[10, 20, 30, 40, 50]

Add (index, value)

```java
    public static void main(String[] args) {

        //declaration
        ArrayList<Integer> arr;
        //initialization
        arr = new ArrayList<>();

        arr.add(10);
        arr.add(20);
        arr.add(30);
        arr.add(1,100);

//          for(int i=0;i<arr.size();i++) {
//              System.out.println(arr.get(i));
//          }

        for(int ele : arr) {
            System.out.println(ele);
        }
    }
}
```

[ ]

[10]

[10, 20]

0 1 2
[10, 20, 30]

0 1 2 3
[10, 100, 20, 30]

Wrapper Class

byte → Byte
short → Short
int → Integer
long → Long
float → float
double → Double
char → Character
boolean → Boolean

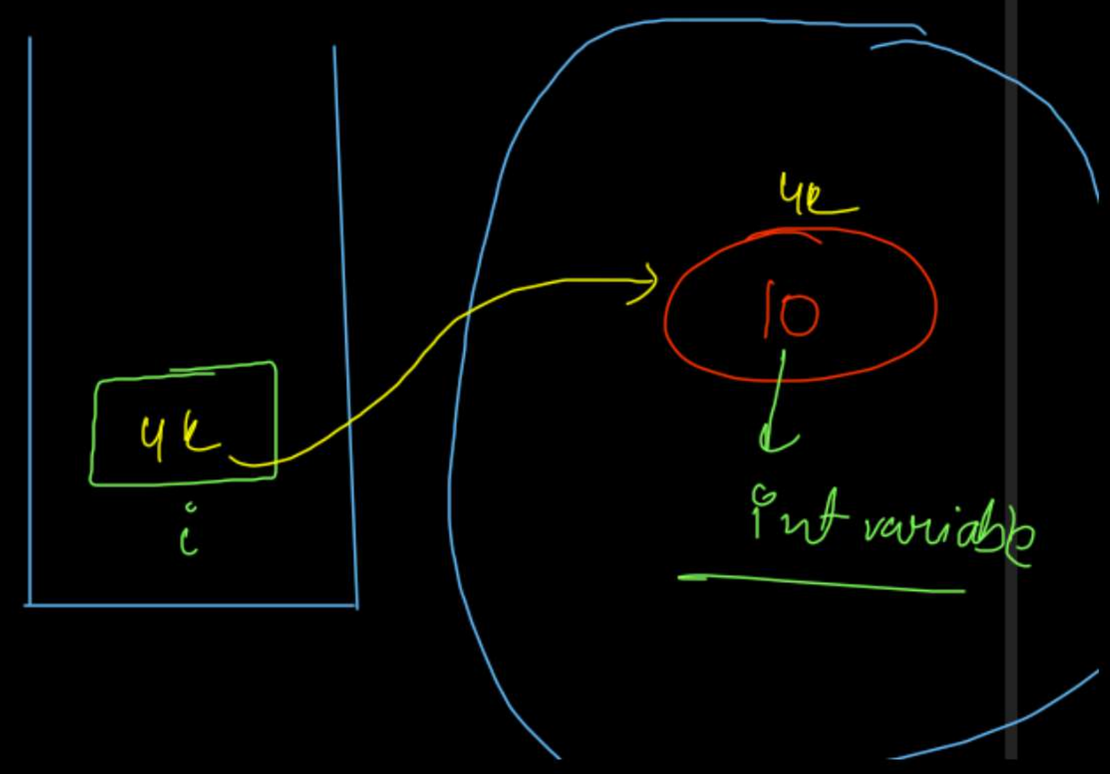
int x = 10;

Integer i = 10;

Integer i = new Integer(10);

↓

autoboxing



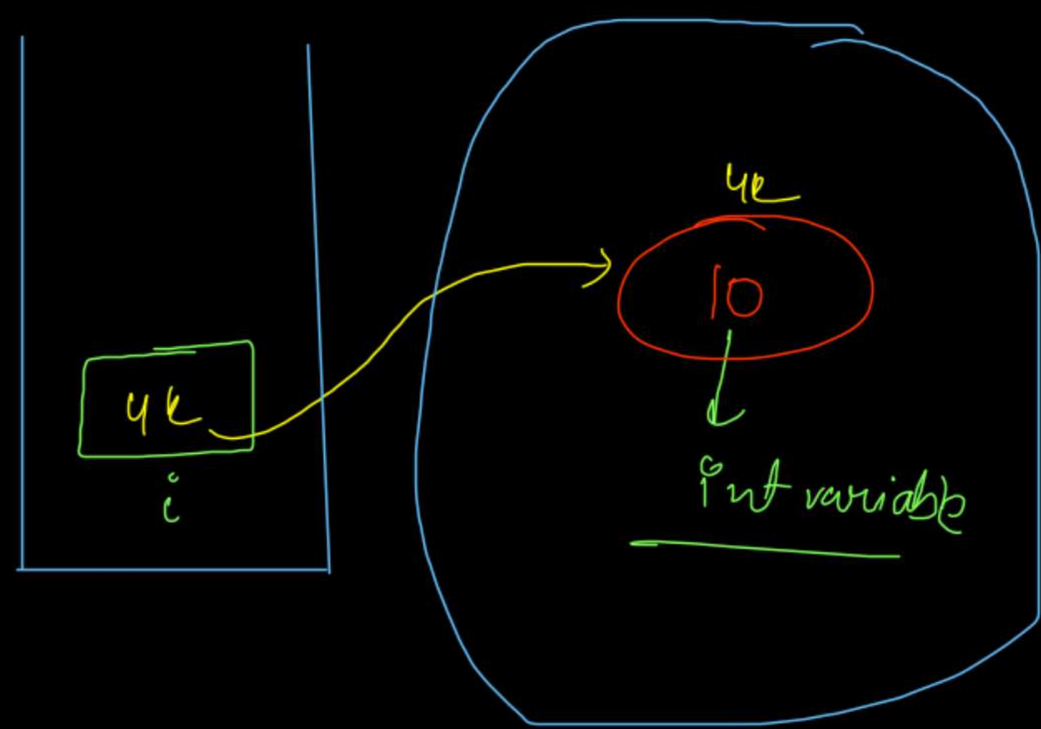
Integer i = 10;

int x = i;

↓

unboxing

(automatically, value is taken from the box)



```java
public static void main(String[] args) {
    Integer i = 10; //autoboxing
    int x = i; //unboxing

    System.out.println(i);
    System.out.println(x);
}
```

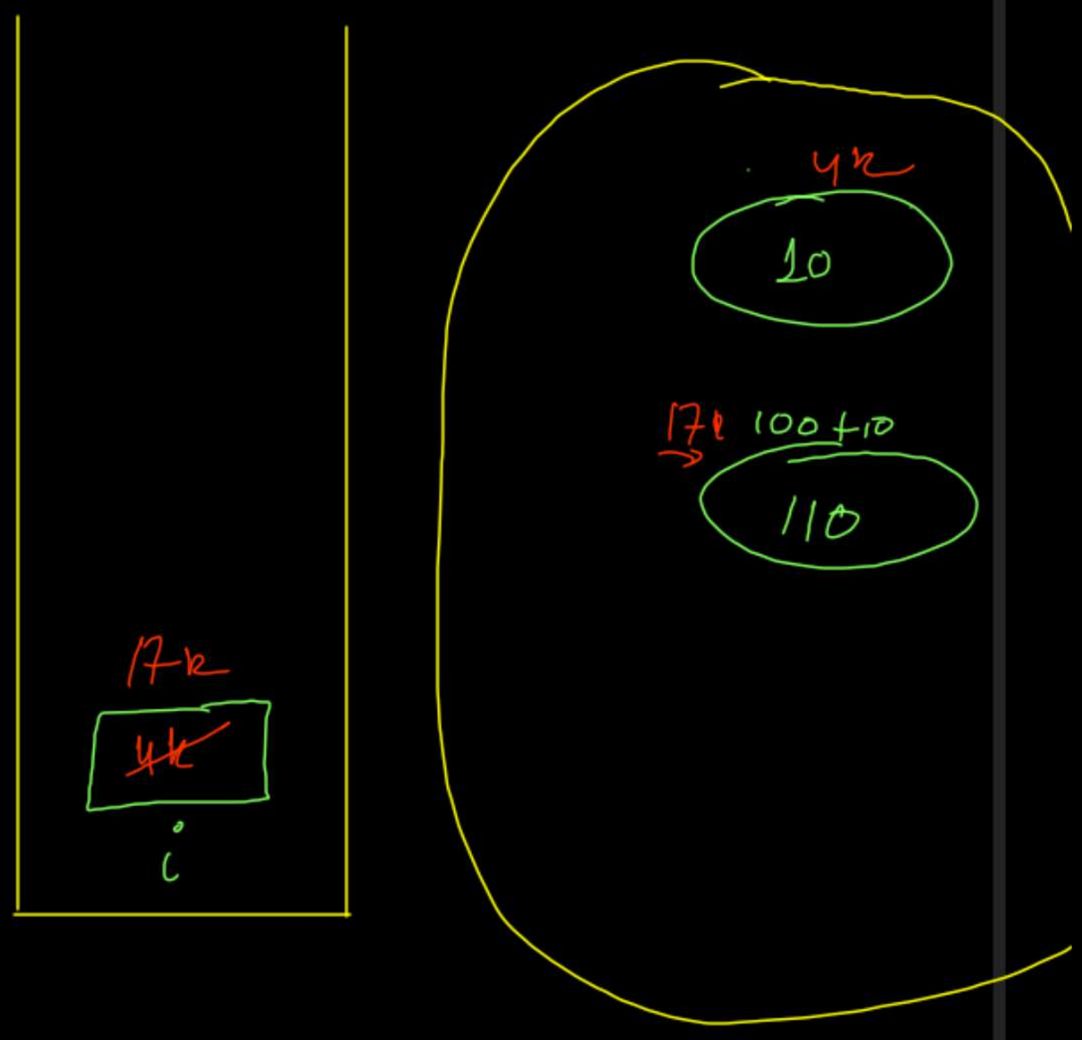
Lecture - 19

- Wrapper Classes (Immutability)
- ArrayList Memory Mapping
- Doubling ++
- ArrayList of ArrayLists (& its Memory Mapping)
- List vs ArrayList

```java
import java.util.*;

public class Main {

    class MyInteger {
        int x;
    }

    public static void main(String[] args) {
        Integer i = 10; //autoboxing
        i += 100;

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

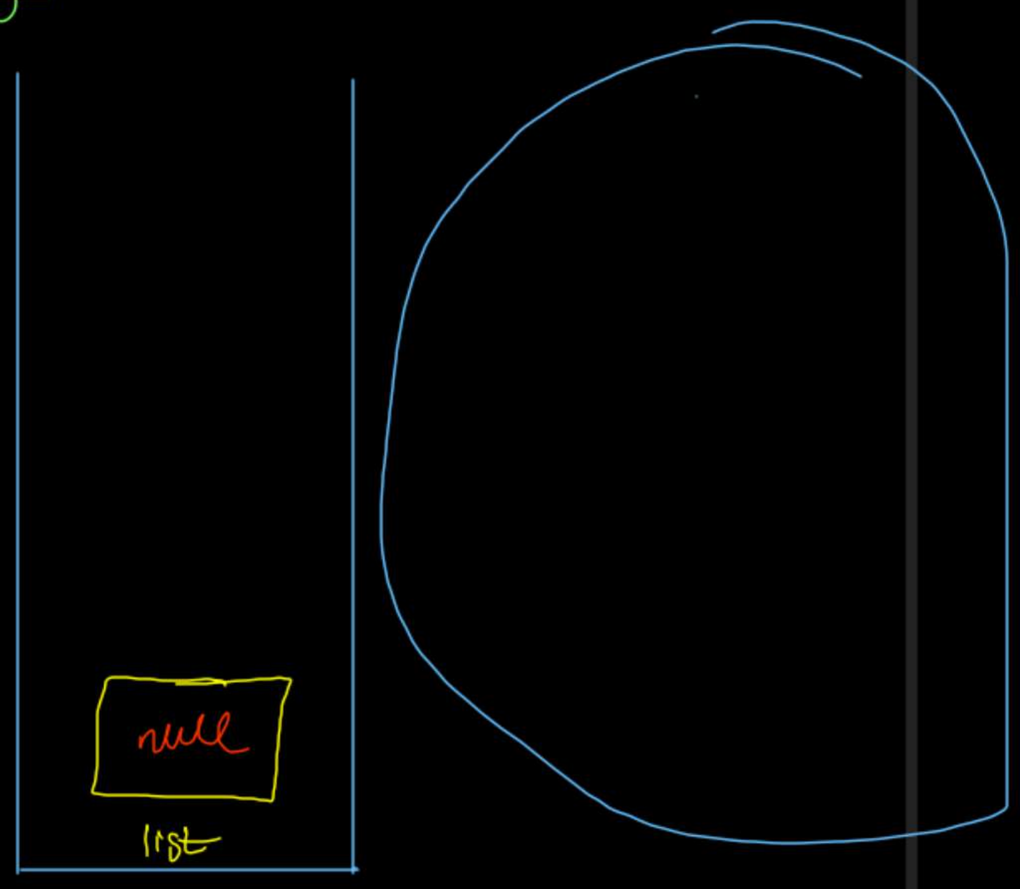
→ data membe

i = i + 100



Arraylist Declaration (Memory Mapping)

```java
import java.util.*;

public class Main {

    class MyInteger {
        int x;
    }

    public static void main(String[] args) {
        ArrayList<Integer> list;
    }
}
```

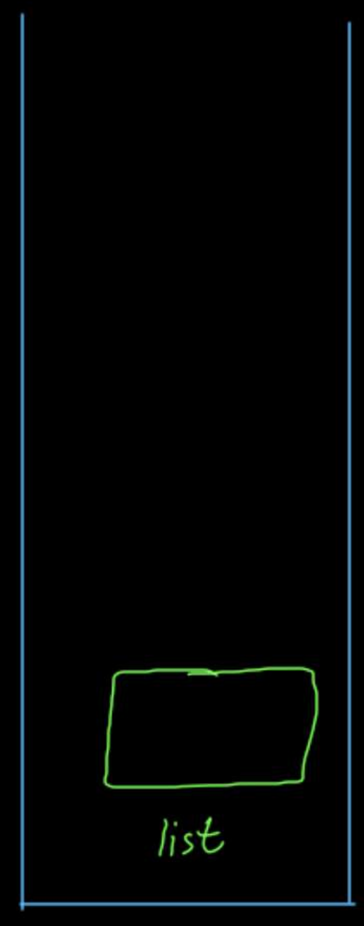


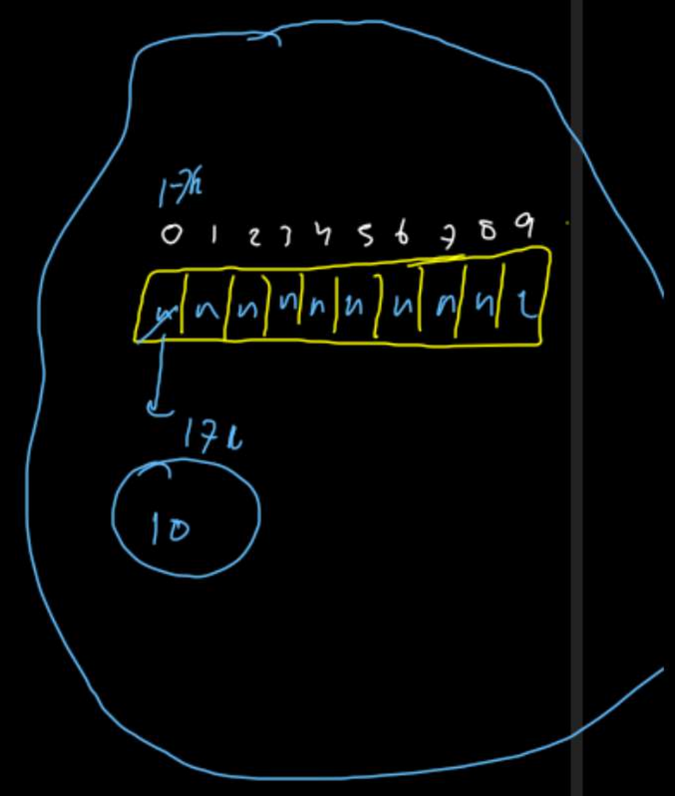
ArrayList Initialization

```java
import java.util.*;

public class Main {

    class MyInteger {
        int x;
    }

    public static void main(String[] args) {
        ArrayList<Integer> list = new ArrayList<>();
    }
}
```

Initial size of AL in Java is 10.

list.add(10)

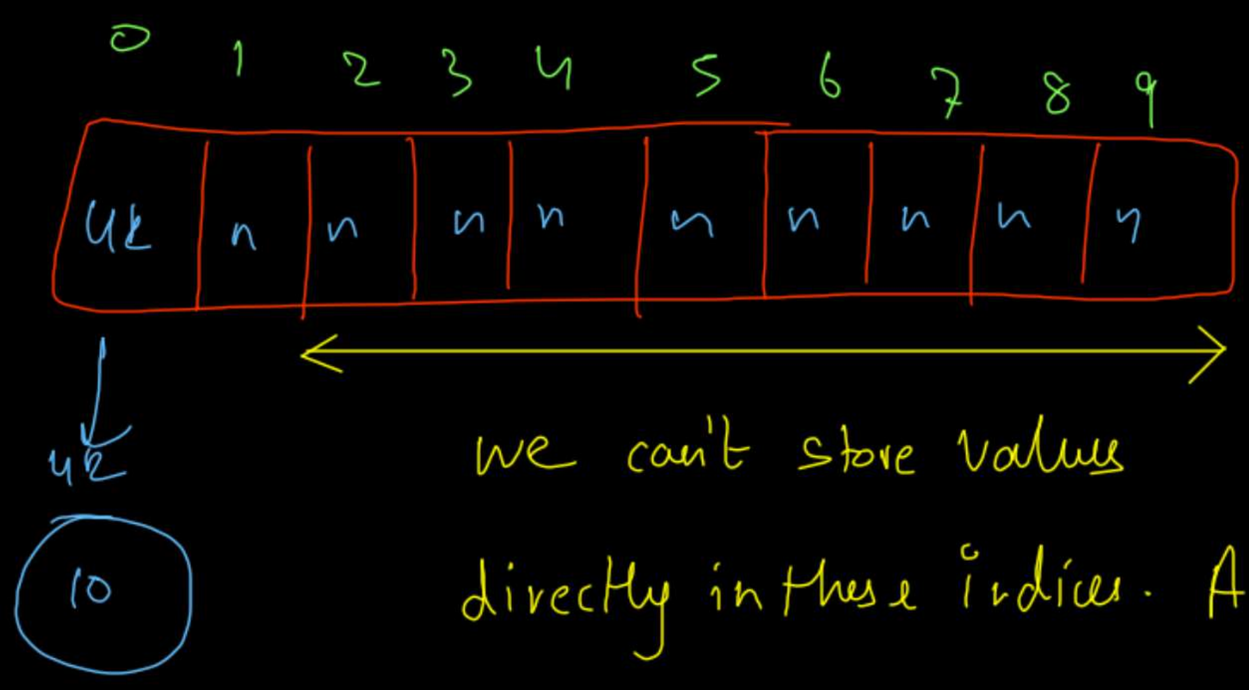




```java
import java.util.*;

public class Main {

    class MyInteger {
        int x;
    }

    public static void main(String[] args) {
        ArrayList<Integer> list = new ArrayList<>();

        for(int i=0;i<10;i++) {
            System.out.println(list.get(i));
        }
    }
}
```

```
Finished in N/A
java.lang.IndexOutOfBoundsException: Index 0 out of bounds for length 0
  at line 64, java.base/jdk.internal.util.Preconditions.outOfBounds
  at line 70, java.base/jdk.internal.util.Preconditions.outOfBoundsCheckIndex
  at line 266, java.base/jdk.internal.util.Preconditions.checkIndex
  at line 359, java.base/java.util.Objects.checkIndex
  at line 427, java.base/java.util.ArrayList.get
  at line 13, Main.main
```

// Even though, there are 10 elements in an AL initially, Java gives us access to only those elements that we have inserted.



0	1	2
10	20	30

we can't store values directly in these indices. Appending a value in the list is allowed just at the index = list.size()

// Concept of Doubling

110

0	1	2	3	4	5	6	7	8	9
10	20	30	40	50	60	70	80	90	100

arr = ~~4k~~ 17k

17k

16

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
10	20	30	40	50	60	70	80	90	100	110	120	130	140	150	160	170	75	7	5

120 130 140 150 160 170 75 7 5

In Java, AL grows by 50% of its original size.

From the Javadoc, it says this is from Java 2 and on, so its a safe bet in the Sun JDK.

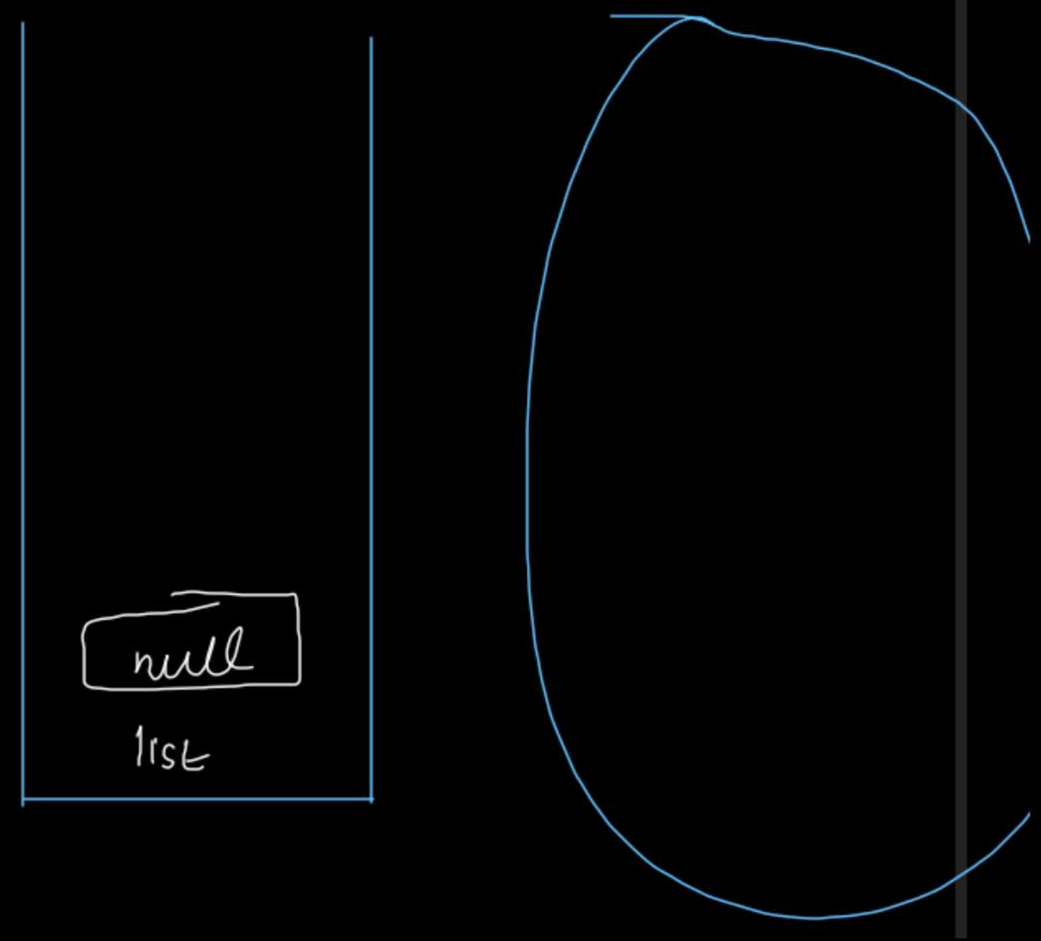
EDIT : for those who didn't get what's the connection between multiplying factor `1.5` and `int newCapacity = oldCapacity + (oldCapacity >> 1);`

`>>` is right shift operator which reduces a number to its half. Thus,

```
int newCapacity = oldCapacity + (oldCapacity >> 1);
=> int newCapacity = oldCapacity + 0.5*oldCapacity;
=> int newCapacity = 1.5*oldCapacity ;
```

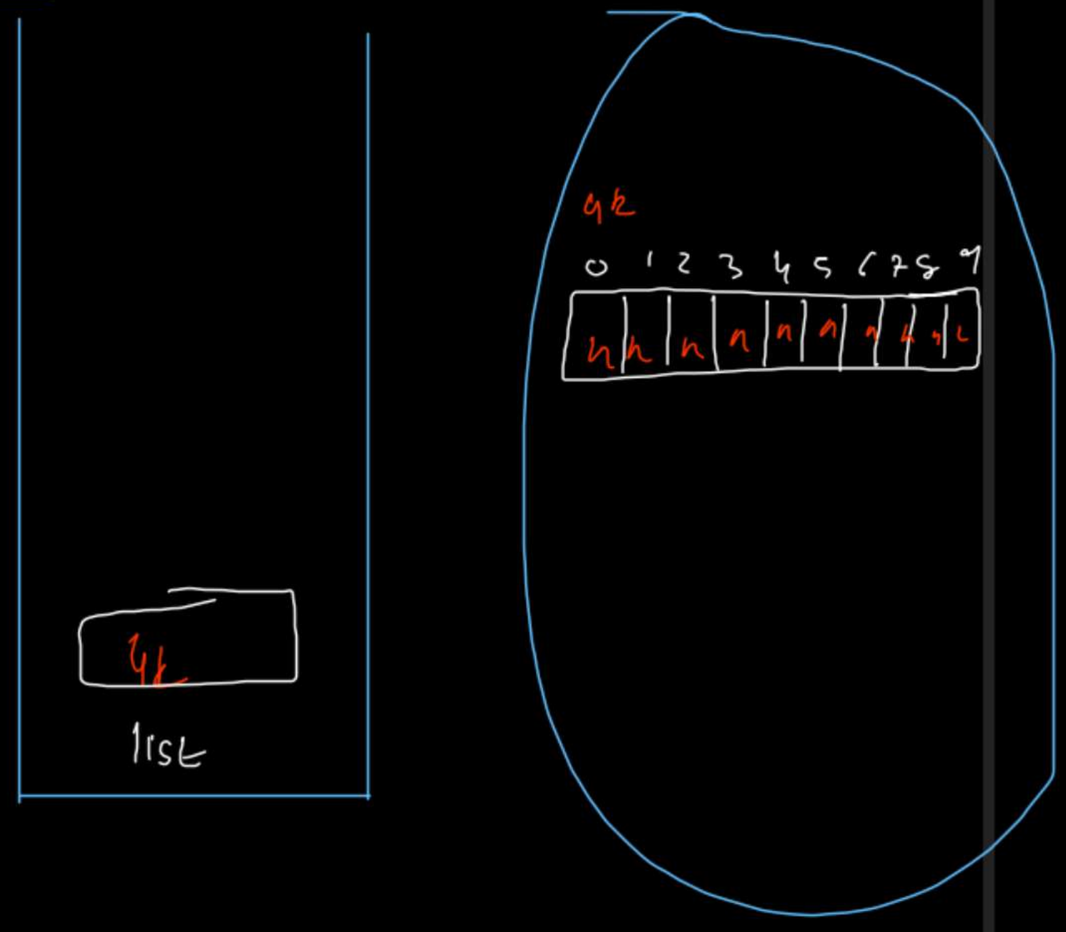
Array List of Array list (Declaration)

```java
import java.util.*;

public class Main {

    class MyIntegerArrayList {
        int[] arr;
    }

    public static void main(String[] args) {
        //declaration
        ArrayList<ArrayList<Integer>> list;
    }
}
```

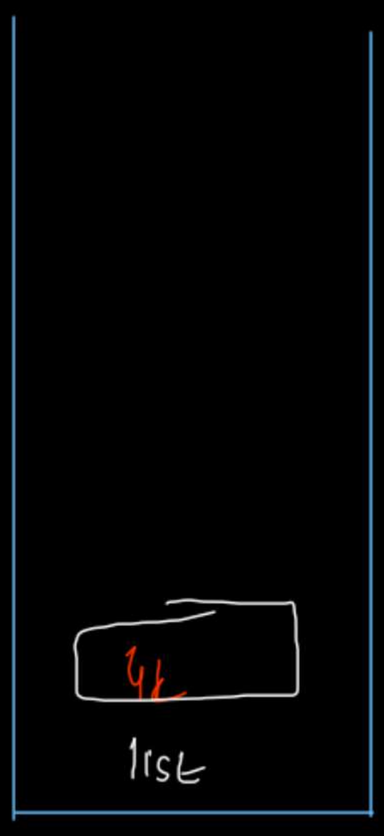


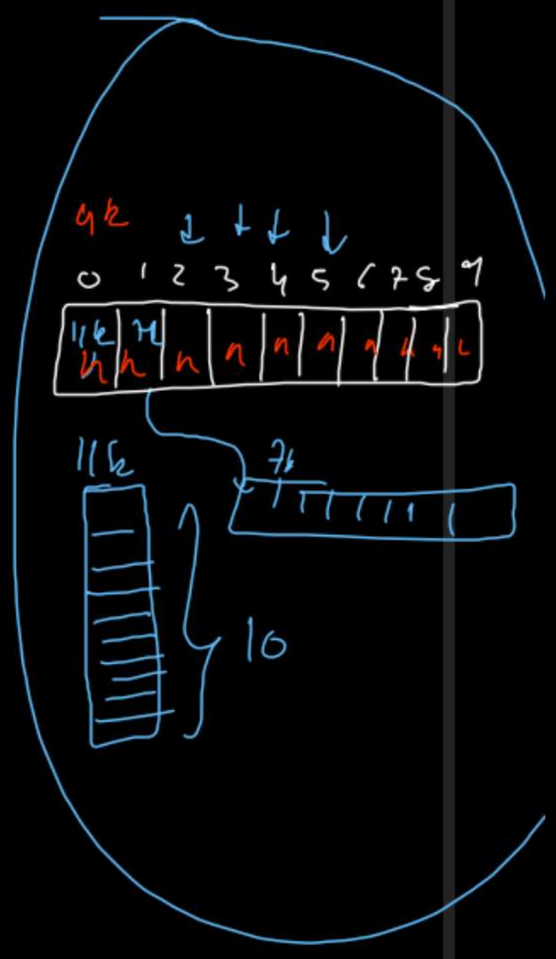
ArrayList of ArrayList (Initialization)

```java
import java.util.*;

public class Main {

    class MyIntegerArrayList {
        int[] arr;
    }

    public static void main(String[] args) {
        //declaration
        ArrayList<ArrayList<Integer>> list = new ArrayList<>();
    }
}
```

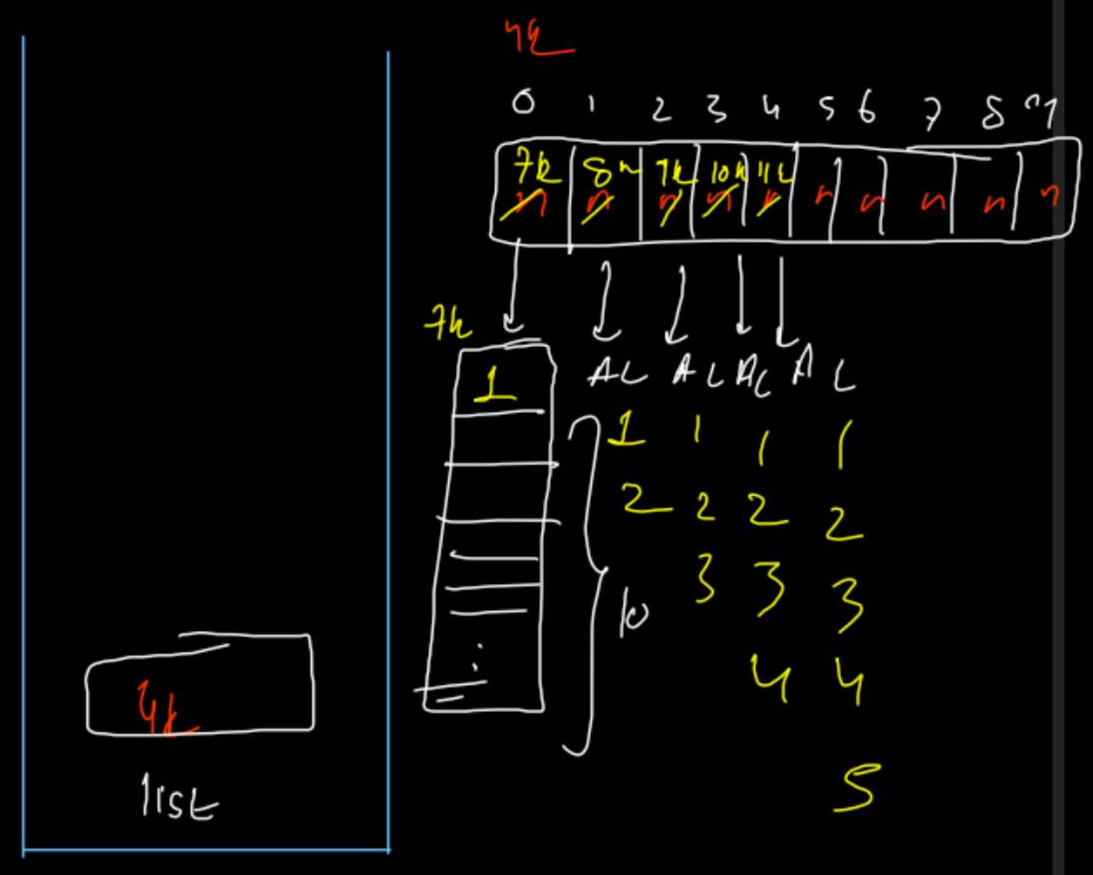


```java
import java.util.*;

public class Main {

    class MyIntegerArrayList {
        int[] arr;
    }

    public static void main(String[] args) {
        //declaration
        ArrayList<ArrayList<Integer>> list = new ArrayList<>();

        for(int i=0;i<5;i++) {
            list.add(new ArrayList<>());
        }


    }
}
```



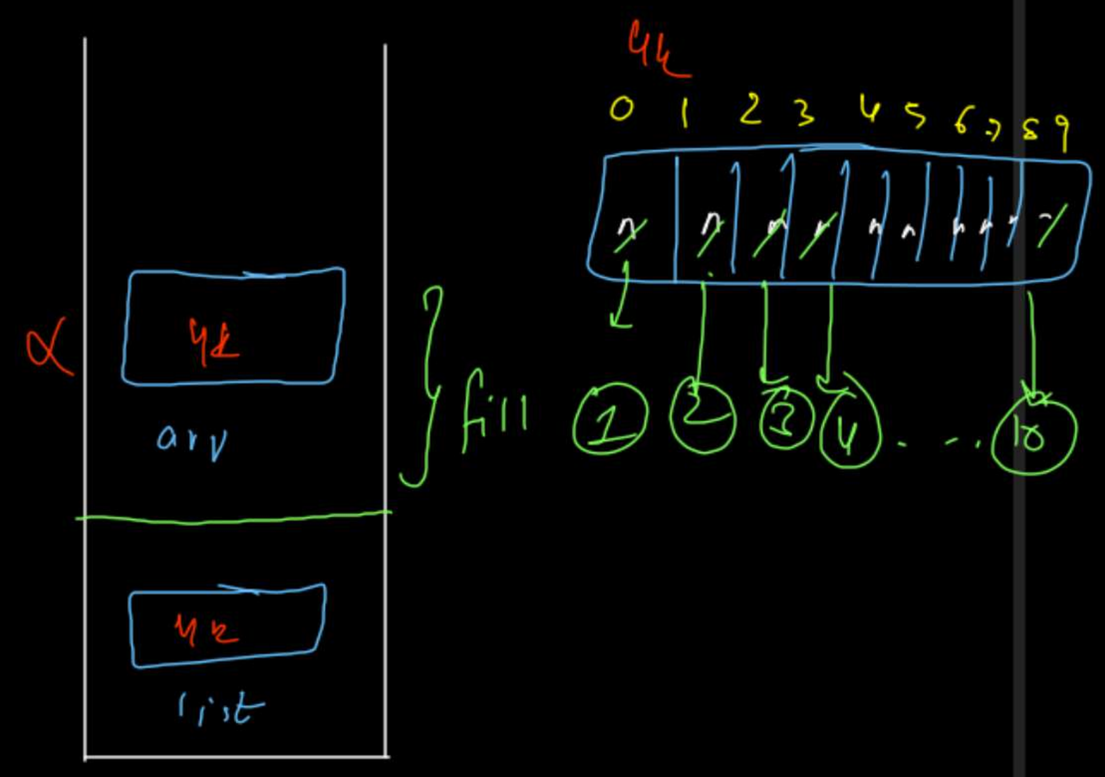


```java
public static void main(String[] args) {
    //declaration
    ArrayList<ArrayList<Integer>> list = new ArrayList<>();

    for(int i=0;i<5;i++) {
        list.add(new ArrayList<>());
    }

    for(int i=0;i<5;i++) {
        for(int j=0;j<=i;j++) {
            list.get(i).add(j+1);
        }
    }

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



ArrayList passed as a function Parameter

```java
import java.util.*;

public class Main {

    public static void fill(ArrayList<Integer> arr) {
        for(int i=1;i<=10;i++) {
            arr.add(i);
        }
    }

    public static void main(String[] args) {
        ArrayList<Integer> list = new ArrayList<>();

        System.out.println("Before calling the fill function " + list);

        fill(list);  //Shallow copy.

        System.out.println("After calling the fill function" + list);
    }
}
```



List vs ArrayList

```
List<Integer> arr = new ArrayList<>();
                or  LinkedList<>();
```

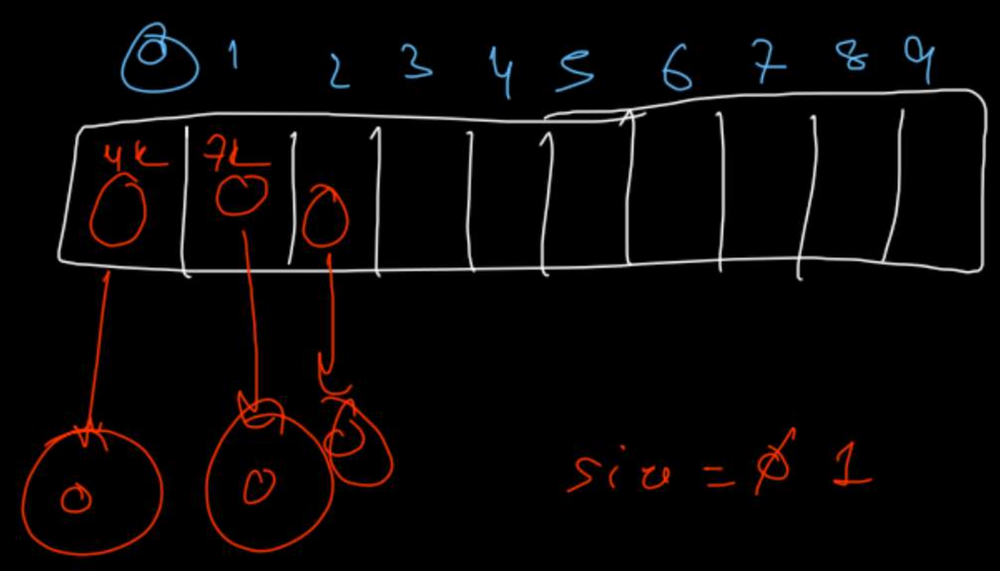
```
ArrayList<Integer> arr = new ArrayList<>();
```

```java
public static void main(String[] args) {
    //Object cannot be of type List
    //because List is an interface and
    //objects of interface cannot be created

    // List<Integer> arr = new List<>();

    List<Integer> arr = new ArrayList<>();
    arr = new LinkedList<>();

    //This is not allowed
    //If the reference is of type arraylist then the object should also
    //be of the type ArrayList
    ArrayList<Integer> list = new LinkedList<>();
    }
}
```

List<Integer> → LC
Arraylist → GfG
} returns

$i = \cancel{1}\ 2$

list.add(0);



size = $\cancel{0}$ 1

Java Collection | Set 1 (ArrayList) Part-1

Basic Accuracy: **50.38%** Submissions: **2157** Points: **1**

ArrayList are dynamic size arrays. Try this problem using ArrayList.

Given a ArrayList of **N** elements and a integer **Q** defining the type of query(which will be either 1 or 2) :
Q = 1 includes two integers **p** and **r**. Which means insert the value r at index p in the ArrayList and print the whole updated ArrayList.
Q = 2 includes one integer **p**. In this query print the index at which the value p is last found in the ArrayList. If the value p is not found in the ArrayList then print "**-1**".

NOTE: Assume 0 based indexing

p = 10

[10] →

[-1] –

0 1 2 3 4 5 6
{10, 20, 10, 30, 10, 40, 50}

```java
ArrayList<Integer> solve(int N, ArrayList<Integer> A, int Q, ArrayList<Integer> Que
{
    // code here
    if(Q == 1) {
        int p = Query.get(0);
        int r = Query.get(1);
        A.add(p,r);
        return A;
    } else {
        ArrayList<Integer> res = new ArrayList<>();
        int p = Query.get(0);
        for(int i=A.size()-1;i>=0;i--) {
            if(A.get(i) == p) {
                res.add(i);
                return res;
            }
        }

        res.add(-1);
        return res;
    }
}
```

Java Collection | Set 1 (ArrayList) Part-2

Easy Accuracy: **46.02%** Submissions: **6083** Points: **2**

Implement different operations on a ArrayList A .

Input:

The first line of input contains an integer **T** denoting the no of test cases . Then T test cases follow. The first line of input contains an integer **Q** denoting the no of queries . Then in the next line are **Q** space separated queries .

A query can be of five types

1. a x (Adds an element x to the ArrayList A at the end)
2. b (Sorts the ArrayList A in ascending order)
3. c (Reverses the ArrayList A)
4. d (prints the size of the ArrayList)
5. e (prints space separated values of the ArrayList)
5. f (Sorts the ArrayList A in descending order)

{9, 8, 6, 3, 4, 7, 2}

↓

{2, 3, 4, 6, 7, 8, 9}

```java
/*sort the ArrayList A in ascending order*/
void sort_ArrayList_Asc(ArrayList<Integer> A)
{
    // Your code here
    Collections.sort(A);
}
```

```java
/*reverses the ArrayList A*/
void reverse_ArrayList(ArrayList<Integer> A)
{
    // Your code here
    Collections.reverse(A);
}
```

{1, 2, 3, 4, 5}

↓ reverse

{5, 4, 3, 2, 1}

{9, 8, 6, 3, 4, 7, 2}

↓ sort (descending order)

→ ① Sort {2, 3, 4, 6, 7, 8, 9}

→ ② Reverse {9, 8, 7, 6, 4, 3, 2}

{9, 8, 7, 6, 4, 3, 2}

Remove Primes

Easy

Prev Next

1. You are given an ArrayList of positive integers.
2. You have to remove prime numbers from the given ArrayList and return the updated ArrayList.

Note -> The order of elements should remain same.

Input Format

A number N
arr1
arr2.. N numbers

i i

$[\overset{0}{10}, \overset{1}{\cancel{13}}, \overset{2}{11}, \overset{3}{17}, \overset{4}{20}, \overset{5}{22}, \overset{6}{23}]$

i

$[\overset{0}{10}, \overset{1}{11}, \overset{2}{17}, \overset{3}{20}, \overset{4}{22}, \overset{5}{23}]$

} Concurrent access & modification anomaly.

Increment i only when we encounter a non-prime no.

```java
public static void solution(ArrayList<Integer> al){
    // write your code here

    int i = 0;
    while(i < al.size()) {
        if(isPrime(al.get(i)) == true) {
            al.remove(i);
        } else {
            i++;
        }
    }

}
```

i

[10, 13, 11, 17, 20, 22, ~~23~~] ① [10, 20, 22]

i i i

[10, 13, 11, ~~17~~, 20, 22]

i

[10, 13, ~~11~~, 20, 22]

i

[10, ~~13~~, 20, 22]

```java
public static void solution(ArrayList<Integer> al){
    // write your code here

    for(int i=al.size() -1;i>=0;i--) {
        if(isPrime(al.get(i)) == true) {
            al.remove(i);
        }
    }

}
```

// Traverse in Reverse

Recursion

PMI (Principle of Mathematical Induction)

PT $\sum_{i=1}^{N} i = \frac{N(N+1)}{2}$ } → Expectation

① Trivial Case (Small Problem) → Base Case

for $i = 1$

$LHS = \sum_{i=1}^{1} i = 1$

$RHS = \frac{1(\cancel{1+1})}{\cancel{2}} = 1$

for $i=1$, $LHS = RHS$

② Assume that LHS = RHS for $i = k$ → faith

$$LHS = \sum_{i=1}^{k} i = 1+2+3+4 \dots + k$$

$$RHS = \frac{k(k+1)}{2}$$

Assume that they are equal

③ for $i = k+1$ → Meeting expectation with faith.

$$LHS = \sum_{i=1}^{k+1} i = 1+2+3+4 \dots + k + (k+1)$$

$$= \frac{k(k+1)}{2} + (k+1) = \frac{(k+1)(k+2)}{2}$$

$$RHS = \frac{N(N+1)}{2} \rightarrow \text{Put } N = k+1$$

$$= \frac{(k+1)(k+1+1)}{2} = \frac{(k+1)(k+2)}{2}$$

LHS = RHS for $i = k+1$

By PMI, this formula holds true.

$\sum_{1}^{N} i$

1 → true

if it is true for k, it will be true for k+1

k k+1

1 2 3 4 . . . . N

k+1

Recursion (Divide & Conquer)

Print Decreasing

① Expectation : printDecreasing(N) → N, N-1, N-2, N-3, ⋮, 1

for N=5: 5, 4, 3, 2, 1

② faith : pd(5) → 5, 4, 3, 2, 1 } pd(4)

pd(4) → 4, 3, 2, 1

pd(N) → N, N-1, N-2, N-3, ⋮, 1 } pd(N-1)

// We have faith that pd(N-1) will do its work.

③ Meeting Expectation with faith:

pd(N) → N ①
pd(N-1) ②

for (N=5)
5 → Print N
4, 3, 2, 1 } pd(4) pd(N-1)

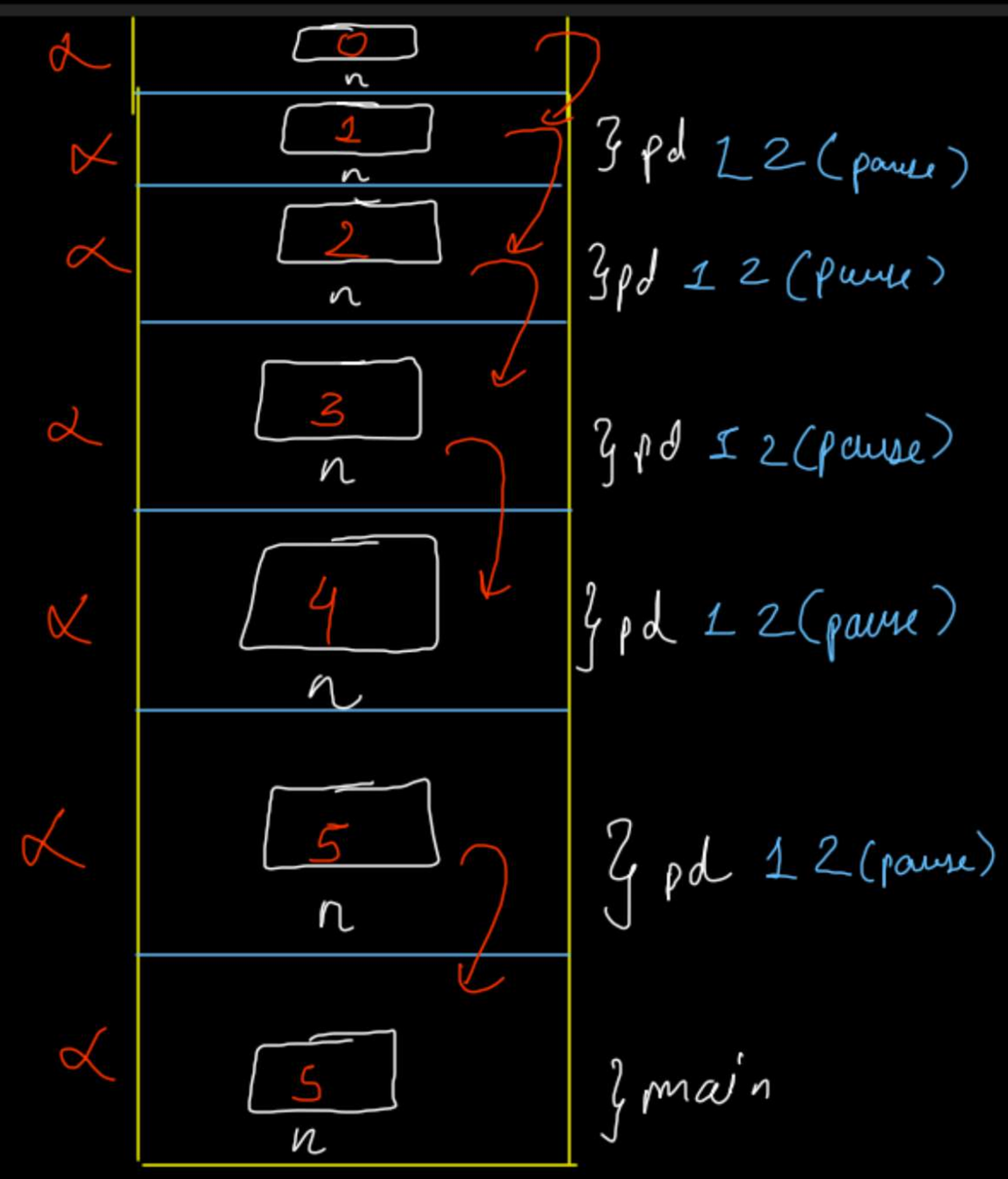
High Level Thinking

```java
public static void main(String[] args) throws Exception {
    // write your code here
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();

    printDecreasing(n);
}

public static void printDecreasing(int n){

    System.out.println(n);
    printDecreasing(n-1);
}
}
```

5
4
3
2
1

if (n == 0) → Base Case
return;

Low-level thinking



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

public class Main {

    public static void main(String[] args) throws Exception {
        // write your code here
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();

        printDecreasing(n);
    }

    public static void printDecreasing(int n){

        if(n == 0) return; //base case

        System.out.println(n);
        printDecreasing(n-1); //recursive call
    }

}
```

Tail Recursion

(when recursive call is the last statement of a recursive function)

Print 1 To N Without Loop

School Accuracy: **51.9%** Submissions: **74255** Points: **0**

Print numbers from **1 to N** without the help of loops.

Example 1:

Input:
N = 10
Output: 1 2 3 4 5 6 7 8 9 10

① Expectation : pi(N) → 1, 2, 3, 4, ⋮, N

pi(5) → 1, 2, 3, 4, 5

② faith : pi(N-1) → 1, 2, 3, 4, ⋮, N-1

pi(4) → 1, 2, 3, 4

③ Meeting expectation with faith

pi(n-1);
Print(n);

pi(4) → 1, 2, 3, 4

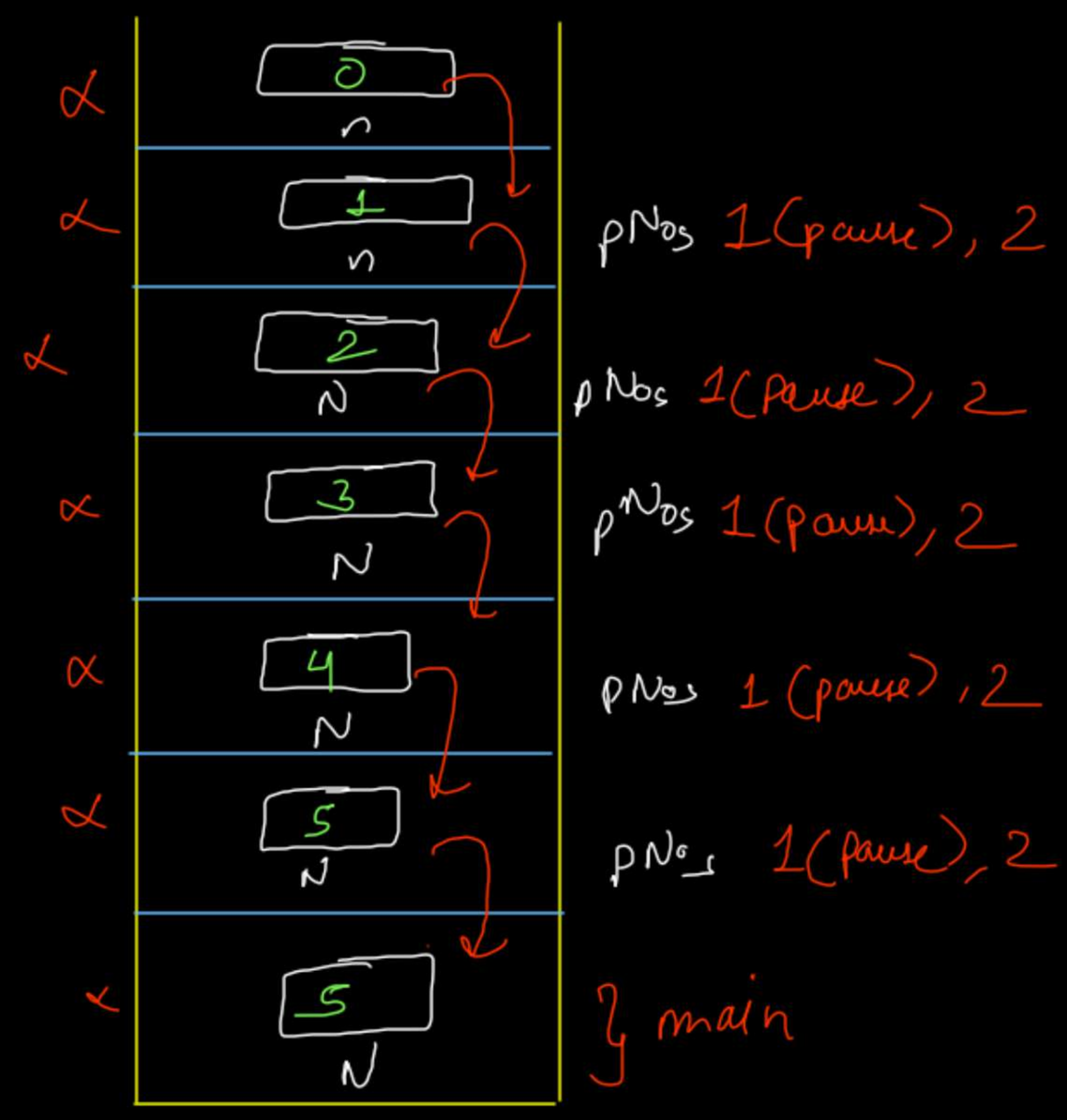
print(5) → 5

```java
class Solution
{

  public void printNos(int N
  {
      //Your code here

    1 printNos(N-1);
    2 System.out.println(N);
  }
}
```

if (n==0) return;

Base Case

1
2
3
4
5



```java
class Solution
{

  public void printNos(int N)
  {
      //Your code here

      if(N == 0) return;

      printNos(N-1);
      System.out.print(N + " ");
  }
}
```

Head Recursion

{ when recursion call is the first statement of a recursive function }

Print Increasing Decreasing

Easy

Prev Next

1. You are given a positive number n.
2. You are required to print the counting from n to 1 and back to n again.
3. You are required to not use any loops. Complete the body of pdi function to achieve it. Don't change the signature of the function.

Note -> The online judge can't force you to write the function recursively but that is what the spirit of question is.Write recursive and not iterative logic. The purpose of the question is to aid learning recursion and not test you.

$N = 3$

3
2
1
1
2
3

① Expectation : pdi(N)

N
N-1
⋮
1
1
2
⋮
N

pdi(3) → 3
2
1
1
2
3

② faith : pdi(N-1) → N-1
N-2
⋮
1
1
2
⋮
N-1

pdi(2) → 2
1
1
2

---

③ Meeting expectation with faith:

Print(N)

pdi(N-1);

Print(N);

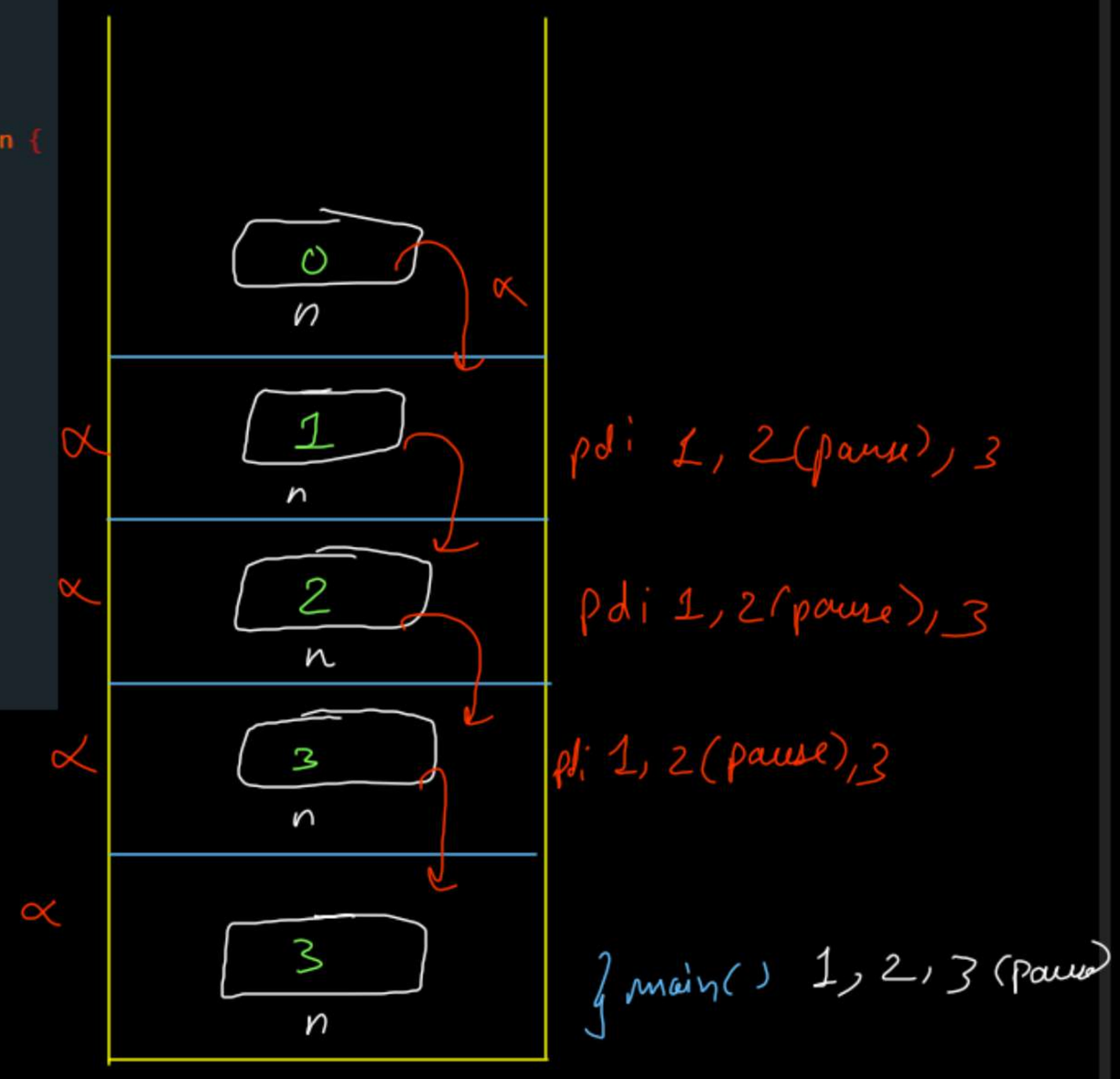
pdi(3) → Print(3)

pdi(2) → 2
1
1
2

Print(3)

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

public class Main {

    public static void main(String[] args) throws Exception {
        // write your code here
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        pdi(n);
    }

    public static void pdi(int n){
        if (n == 0) return;
        System.out.println(n); 1

        //recursion call
        pdi(n-1); 2

        System.out.println(n); 3
    }

}
```

3
2
1
1
2
3



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

public class Main {

    public static void main(String[] args) throws Exception {
        // write your code here
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        pdi(n);
    }

    public static void pdi(int n){

        if(n == 0) return;

        System.out.println(n);

        //recursion call
        pdi(n-1);

        System.out.println(n);
    }

}
```

Factorial

Basic Accuracy: **53.69%** Submissions: **35842** Points: **1**

Given a positive integer, **N**. Find the factorial of **N**.

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

or

$5 \times 4 \times 3 \times 2 \times 1$

$N! = N * (N-1) * (N-2) \ldots\ldots 1$

① Expectation : factorial (N) → N * (N-1) * (N-2) . . . 1

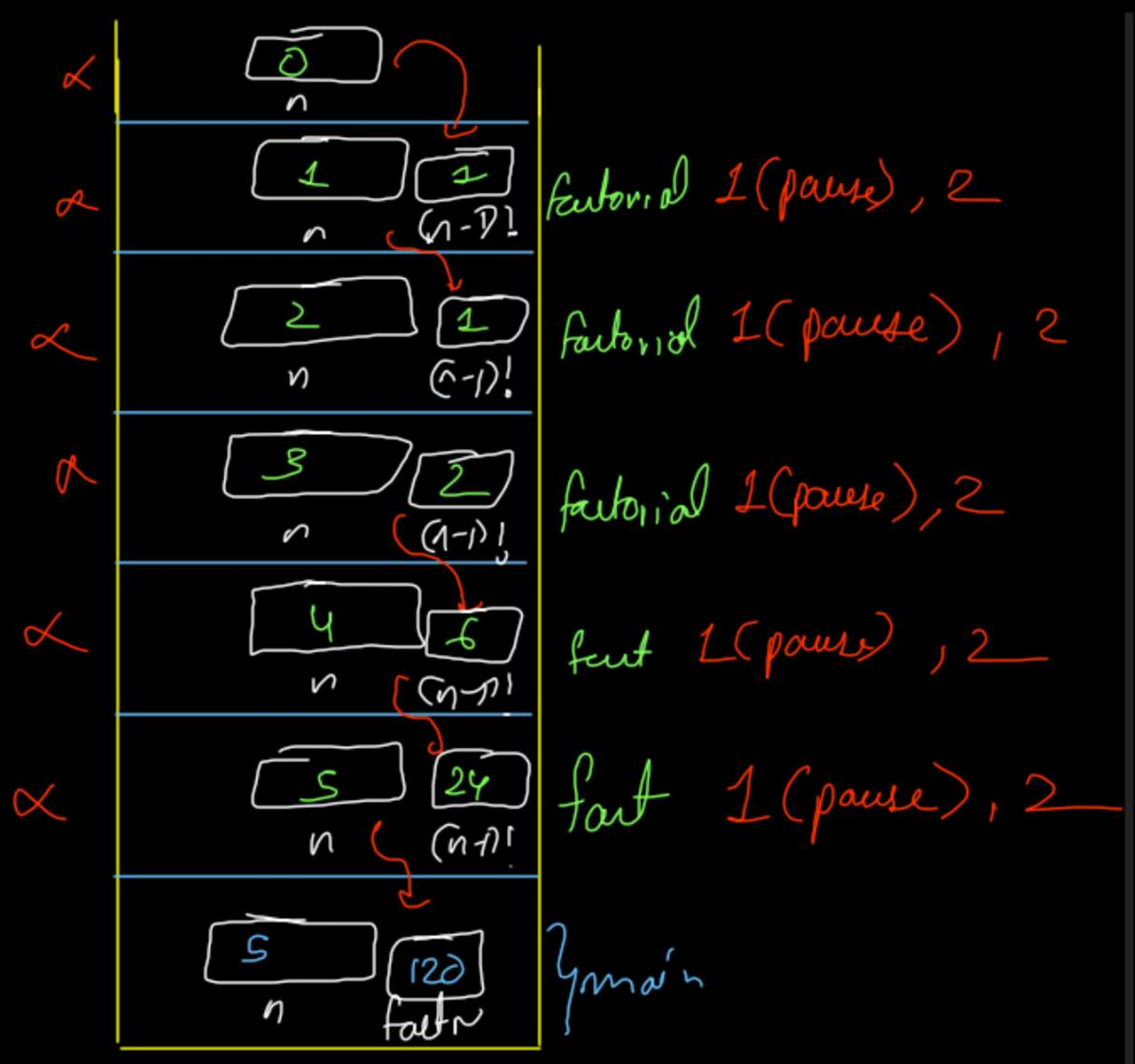
② faith : factorial (N-1) → (N-1) * (N-2) * (N-3) 1

③ Meeting expectation with faith : N * factorial (N-1)

```
class Solution{
    static long factorial(int N){
        long nm1Factorial = factorial(N-1);
        return N * nm1Factorial;
    }
}
```

(1) 1 2

if (n==0) return 1;



```java
class Solution{
    static long factorial(int N){
        // code here
        if(N == 0) return 1L;

        long nm1Factorial = factorial(N-1);
        return N * nm1Factorial;
    }
}
```

Print Zigzag

Easy

Prev Next

1. Here are a few sets of inputs and outputs for your reference
Input1 -> 1
Output1 -> 1 1 1

Input2 -> 2
Output2 -> 2 1 1 1 2 1 1 1 2

Input2 -> 3
Output3 -> 3 2 1 1 1 2 1 1 1 2 3 2 1 1 1 2 1 1 1 2 3

2. Figure out the pattern and complete the recursive function pzz to achieve the above for any positive number n.

Note -> The online judge can't force you to write the function recursively but that is what the spirit of question is.Write recursive and not iterative logic. The purpose of the question is to aid learning recursion and not test you.

Expectation

```
3211121112321112111123
```

→ for N = 3

faith :

```
211121112
```

→ for N = 2

Meeting Expec : With faith

```
3211121112321112111123
```

① Print N

② pzz(N-1)

③ Print(N)

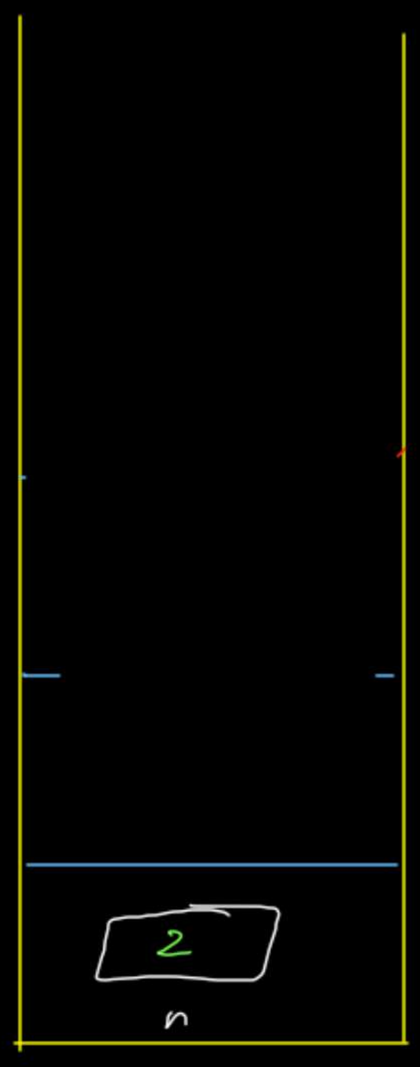
④ pzz(N-1)

⑤ Print(N)

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

public class Main {

    public static void main(String[] args) throws Exception {
        // write your code here
    }

    public static void pzz(int n){

        System.out.print(n + " "); 1
        pzz(n-1); 2
        System.out.print(n + " "); 3
        pzz(n-1); 4
        System.out.print(n + " ");5
    }

}
```

2
1
1
1
1
2
1

1
1
2



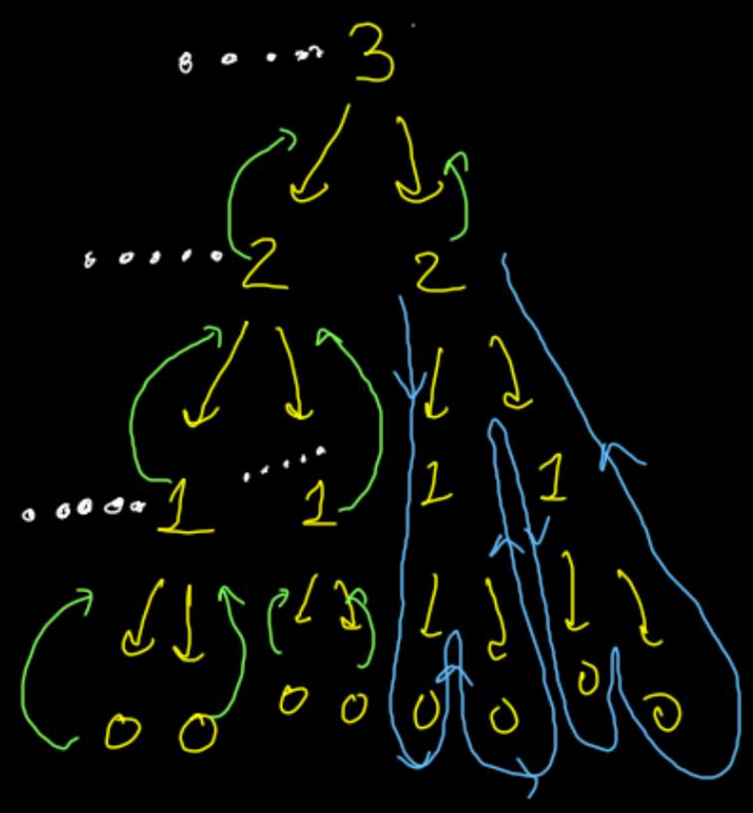
pzz 1, 2(p), 3, 4, 5

& main

Recursion Tree (Euler Tree)

3 2 1 1 1 2 1 1 1 2 3 [2 1 1 1 2 1 1 1 2] 3

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

public class Main {

    public static void main(String[] args) throws Exception {
        // write your code here
    }

    public static void pzz(int n){

        System.out.print(n + " ");
        pzz(n-1);
        System.out.print(n + " ");
        pzz(n-1);
        System.out.print(n + " ");
    }

}
```




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

public class Main {

    public static void main(String[] args) throws Exception {
        // write your code here
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        pzz(n);
    }

    public static void pzz(int n){

        if(n == 0) return;

        System.out.print(n + " ");
        pzz(n-1);
        System.out.print(n + " ");
        pzz(n-1);
        System.out.print(n + " ");
    }

}
```

factorial (Recursion Tree)

5! (120)

↓

(24) 4! * 5

↓

(6) 3! * 4

↓

(2) 2! * 3

↓

(1) 1! * 2

↓

0! * 1

(1)

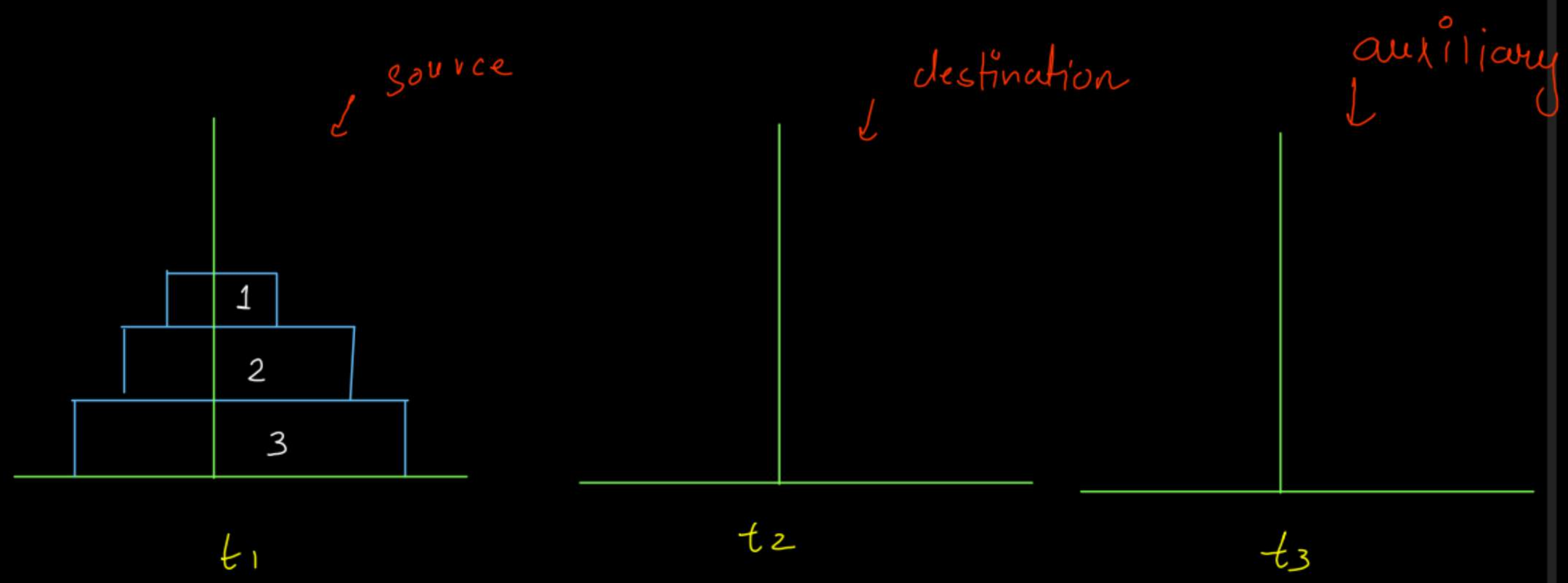
Lecture - 21

Tower of Hanoi Problem

- Understanding the Problem Statement
- High Level Thinking
- Program (With Base Case)
- Low level thinking → Using Call Stack
- Low level thinking → Using Euler Tree
- Conclusion

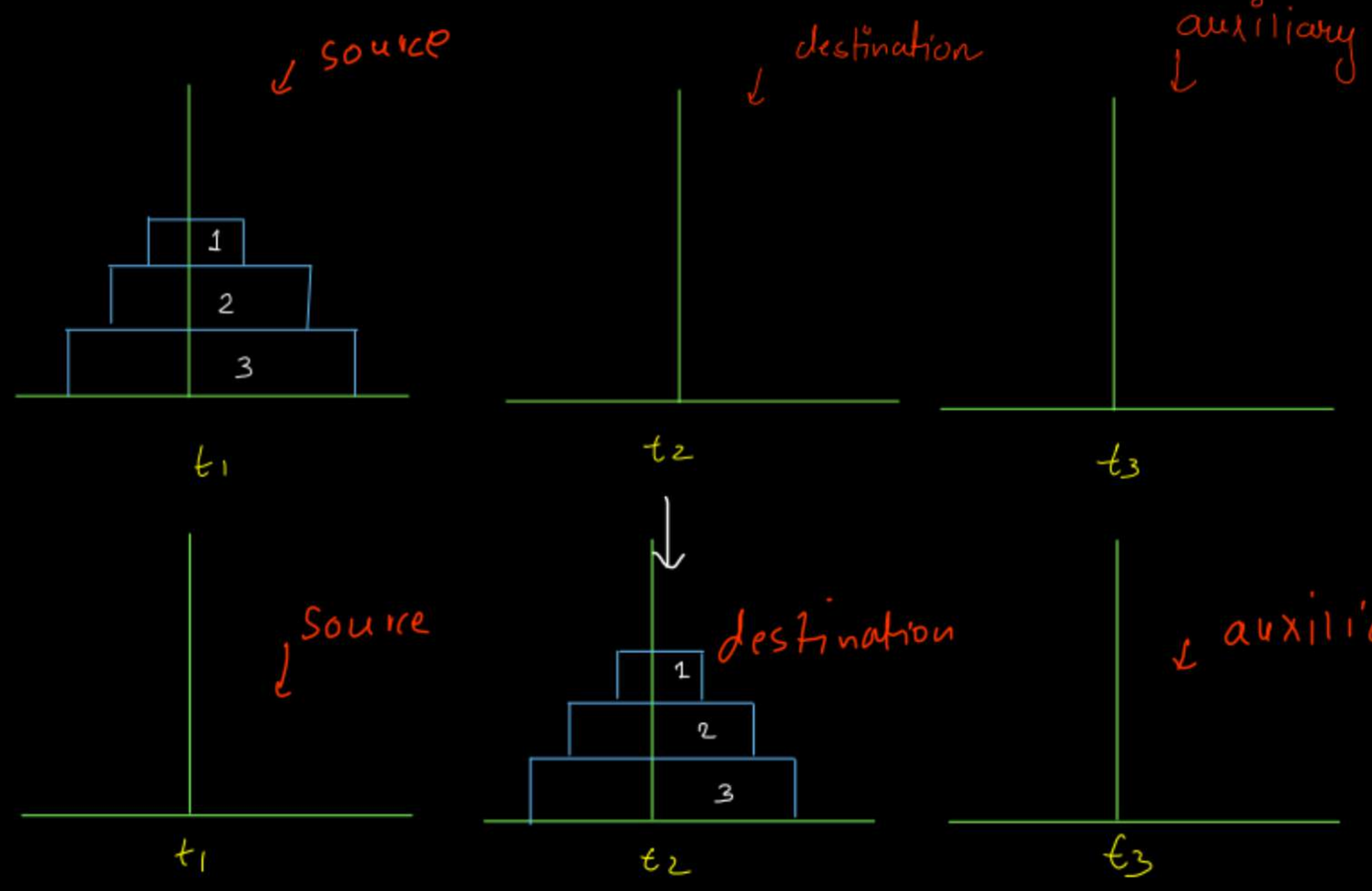
Toh Problem

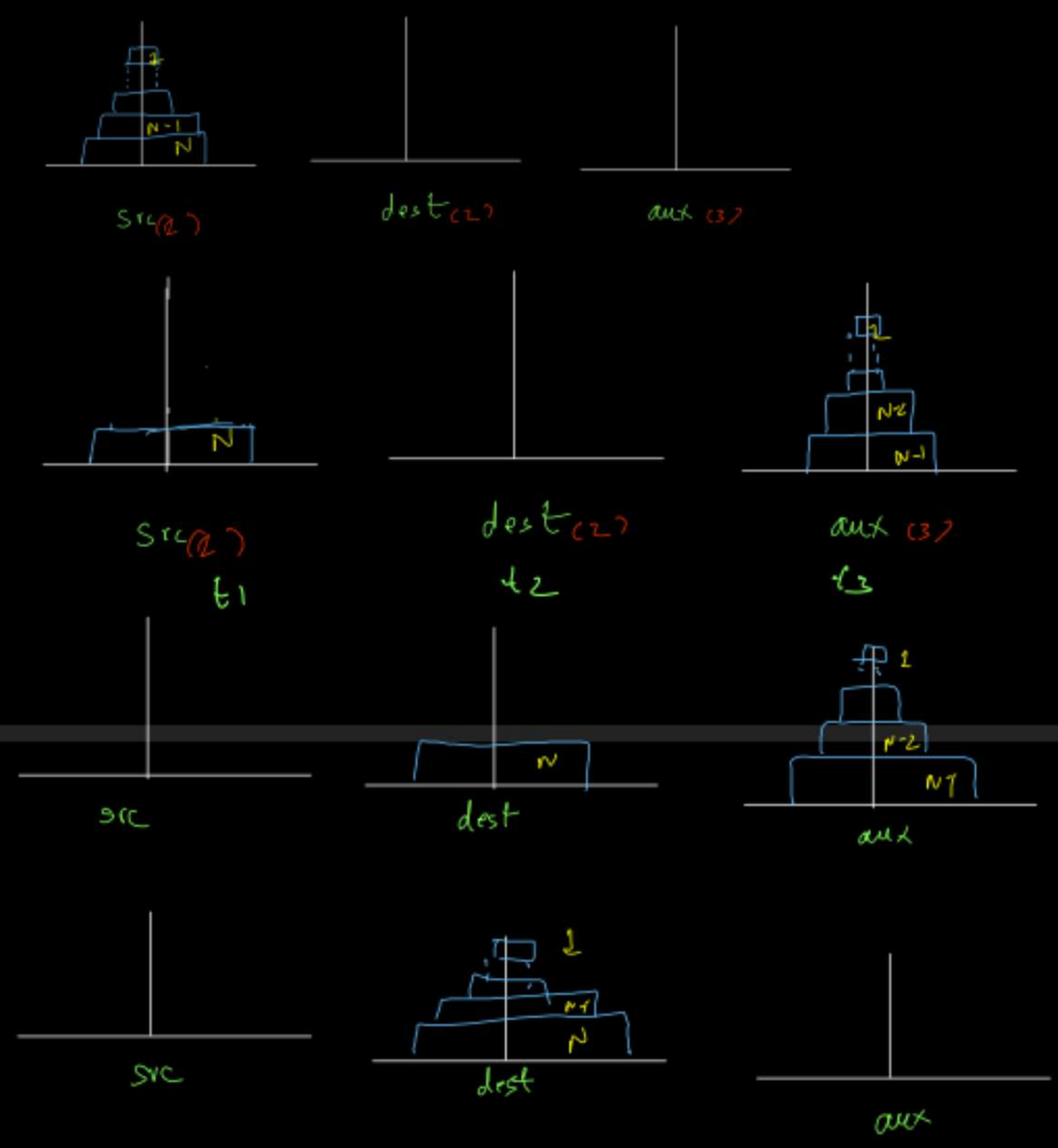
3.3. following the rules
  3.3.1 move 1 disk at a time.
  3.3.2 never place a smaller disk under a larger disk.
  3.3.3 you can only move a disk at the top.



Expectation (for N=3) toh(N, t_1, t_2, t_3)

** following the rules





src dest aux

$toh(N-1, t_1, t_3, t_2)$ {faith 1}

$x \rightarrow$ no of moves

src dest aux

$toh(1, t_1, t_2, t_3)$ 1

$x + y + 1$

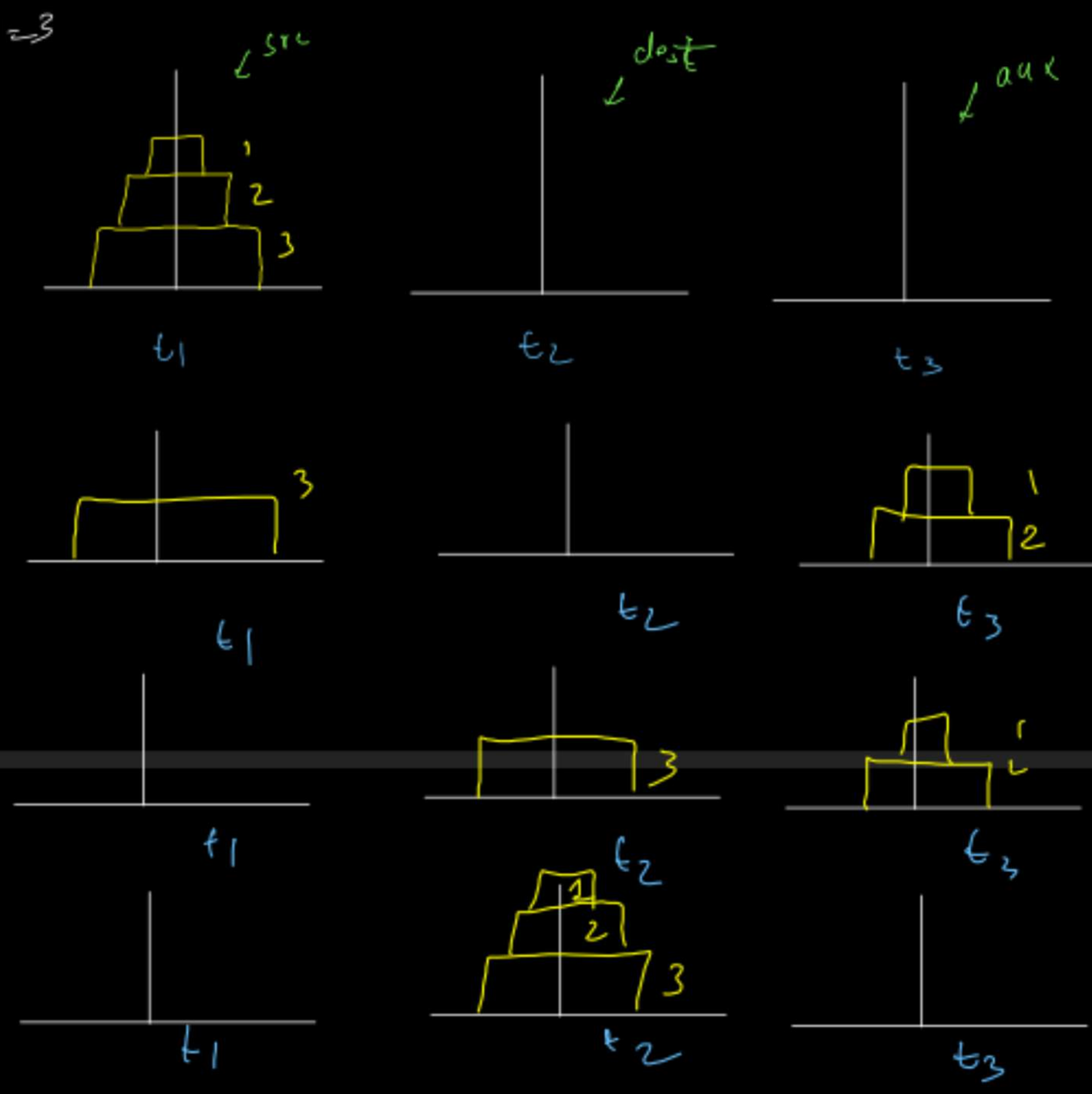
& $x = y$

src dest aux

$toh(N-1, t_3, t_2, t_1)$ {faith 2}

$y \rightarrow$ No of moves.

$N = 3$



$2 * x + 1$

src dest aux

$\underbrace{toh(N-1, t_1, t_3, t_2)}_{x}$

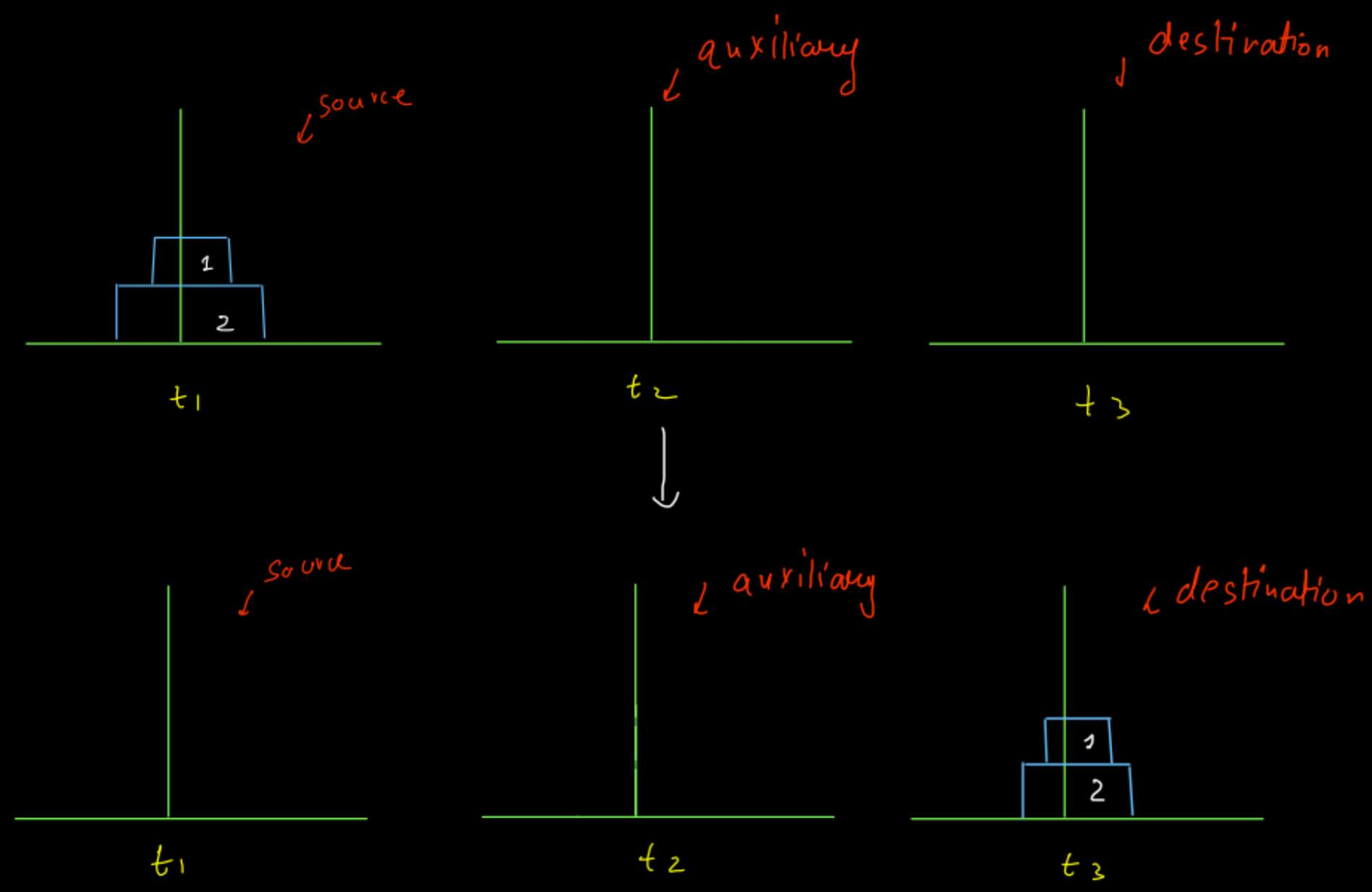
$N^{th} \rightarrow t_1$ to t_2

1

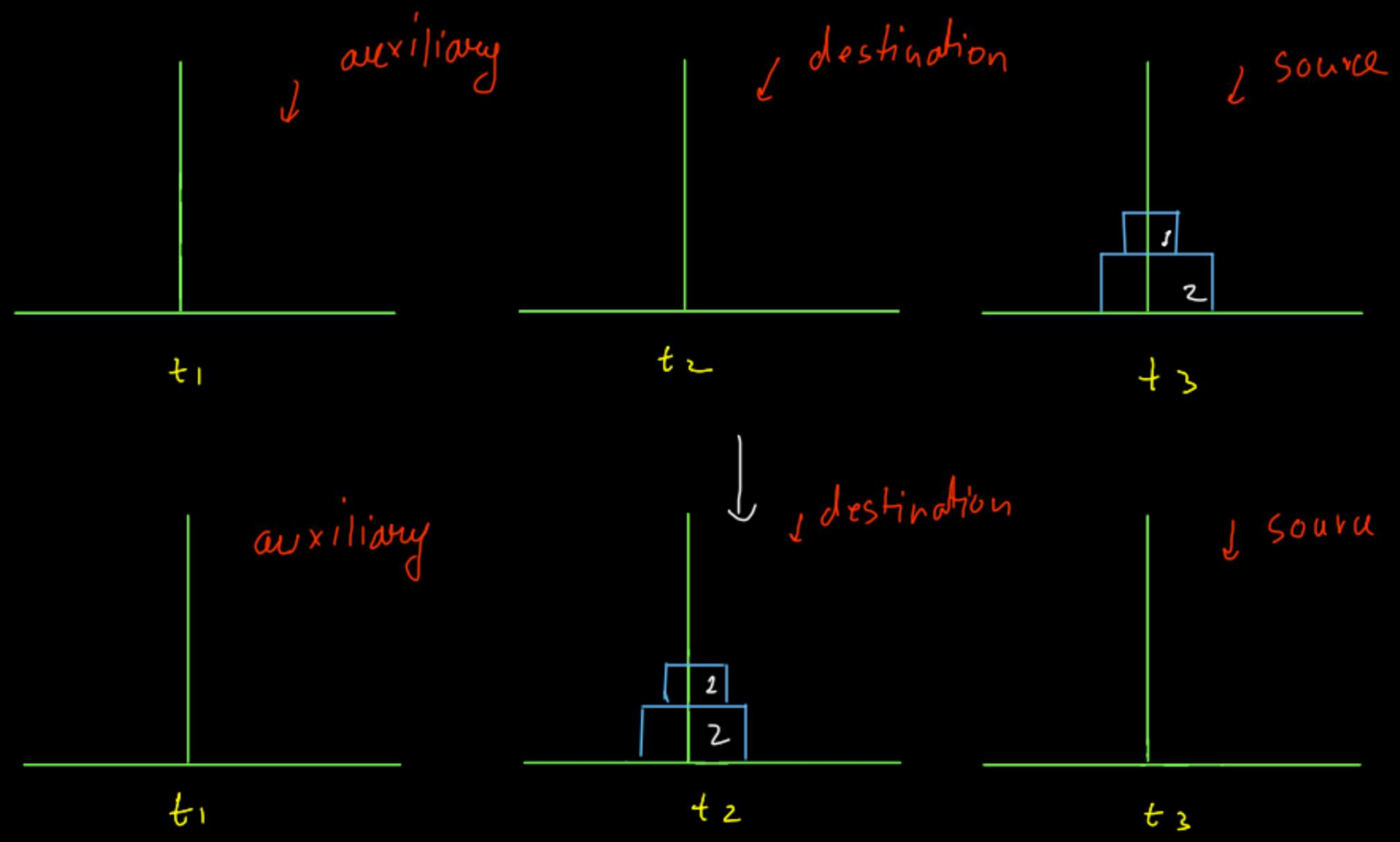
src dest aux

$\underbrace{toh(N-1, t_3, t_2, t_1)}_{x}$

faith1 (N = 2) toh(N-1, t1, t3, t2)

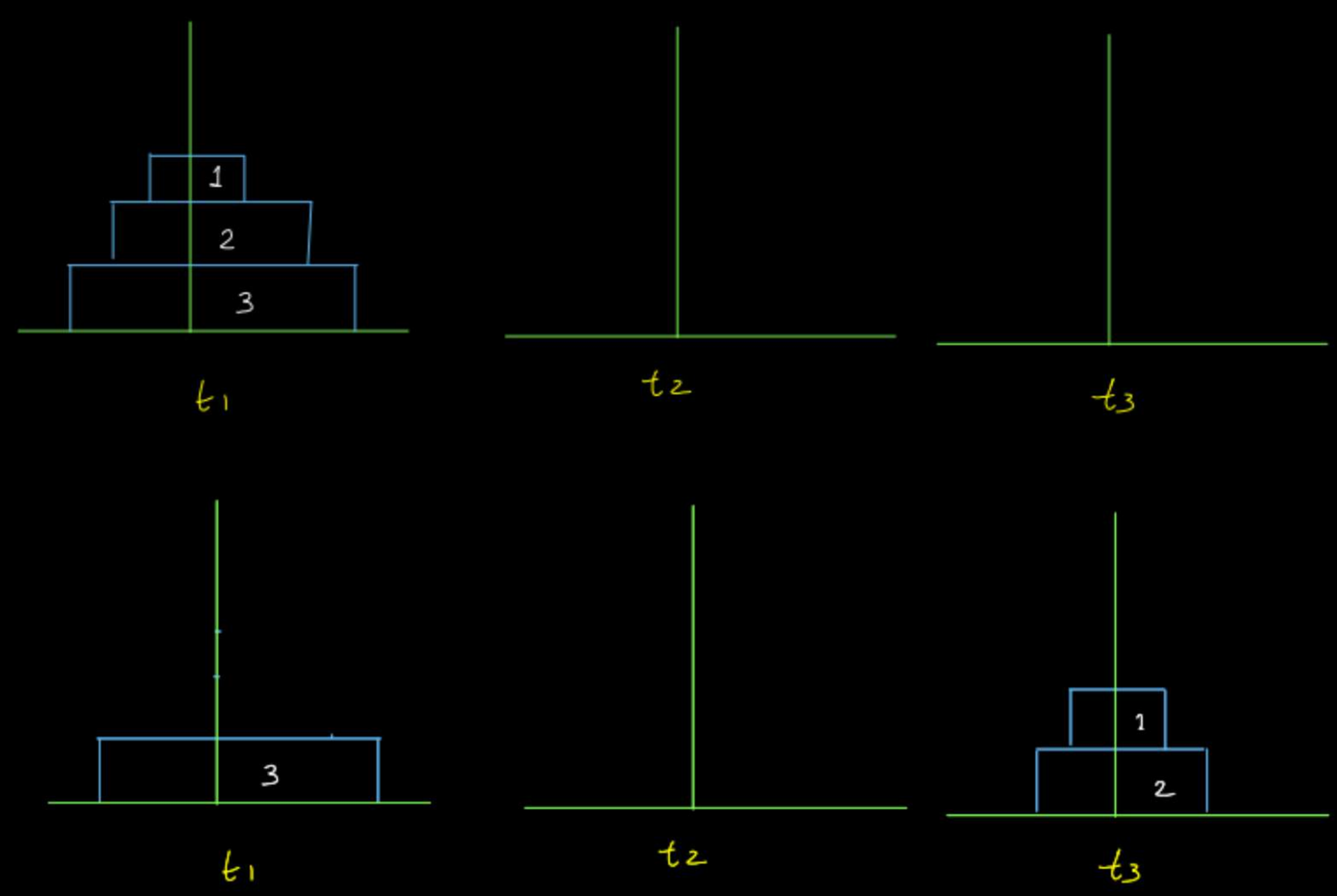


faith 2 (N = 2) toh (N-1, t3, t2, t1)



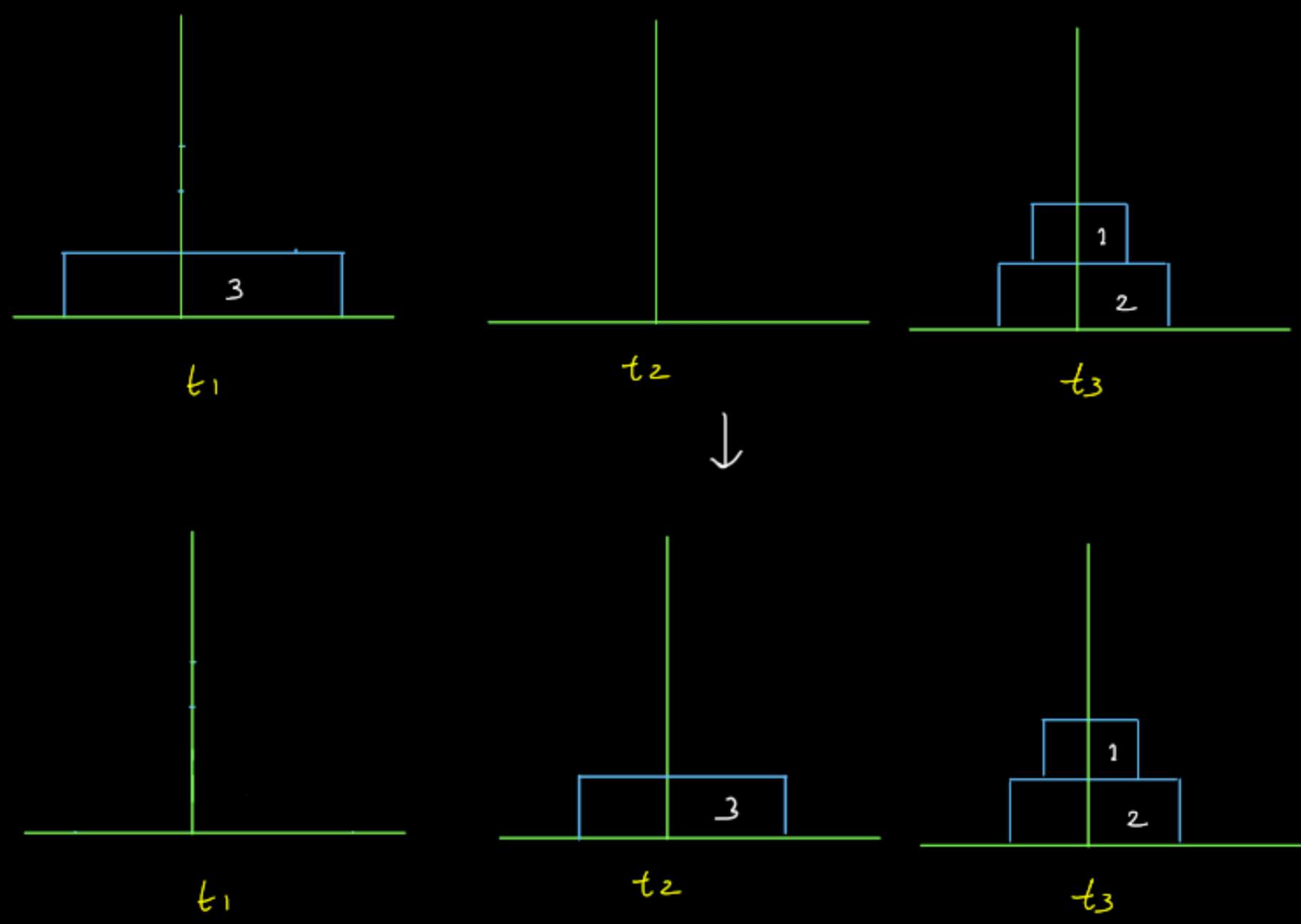
Meeting Expectation with faith

① Move (N-1) disks from t_1 to t_3 using t_2. (faith 1)



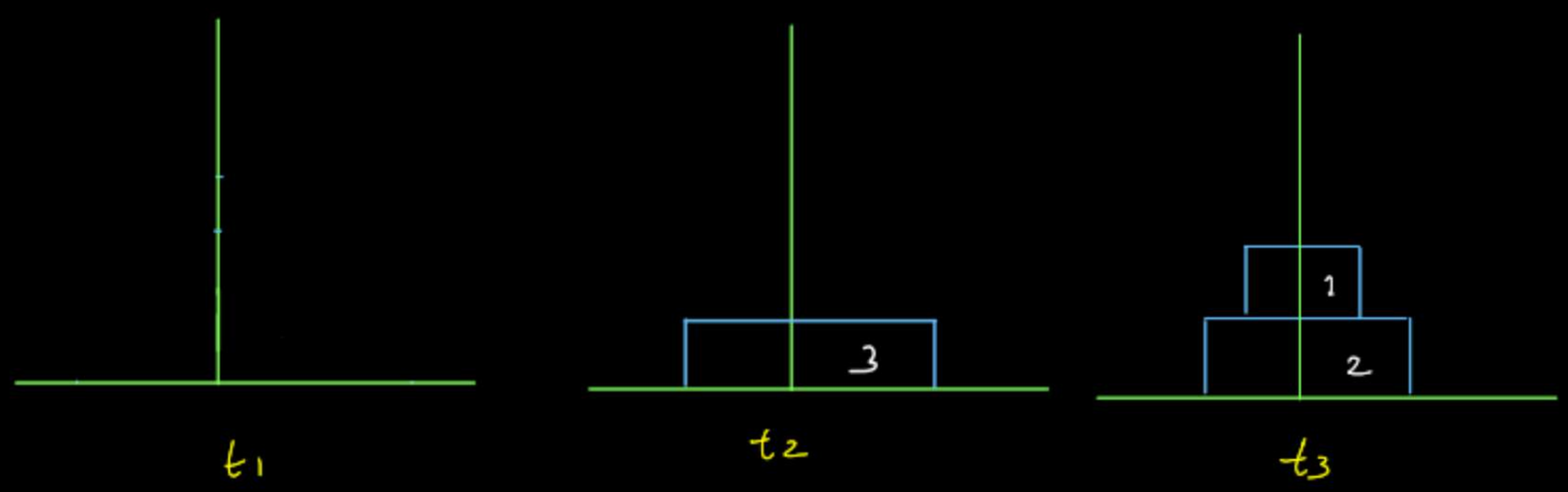
Meeting Expectation with faith

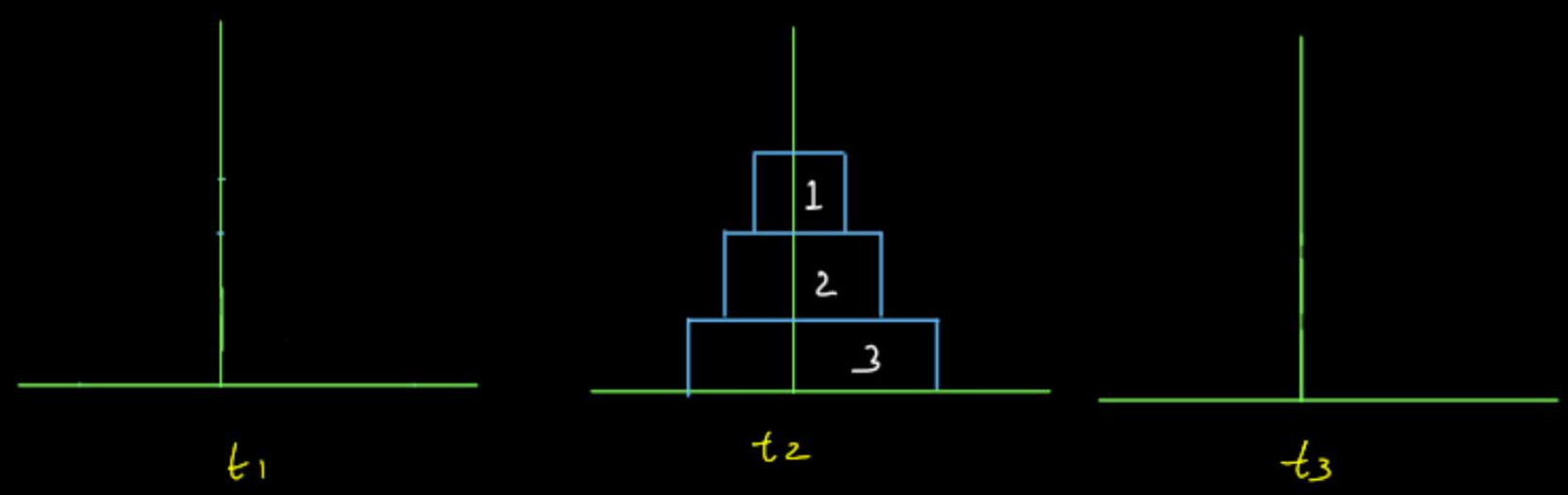
② Move N^{th} disk from t_1 to t_2.



Meeting Expectation with faith

③ Move (N-1) disks from t3 to t2 using t1. (faith 2)

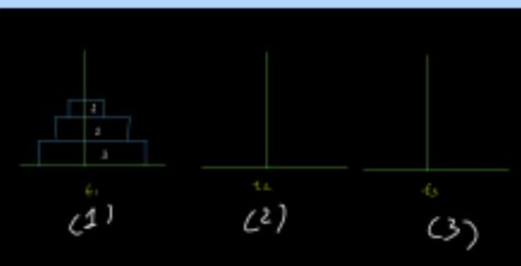




N = 3

```java
// t1 -> src t2-> dest t3-> aux
public long toh(int N, int t1, int t2, int t3) {
    // Your code here

    if(N == 0) return 0;

    long ans1 = toh(N-1,t1,t3,t2);
    System.out.println("move disk " + N + " from rod " + t1 + " to rod " + t2);
    long ans2 = toh(N-1,t3,t2,t1);

    return ans1 + ans2 + 1;
}
```

1 2 3 4



Check recording to view call stack dry run

1. from 1 to 2
2. from 1 to 3
1. from 2 to 3
3. from 1 to 2
1. from 3 to 1
2. from 3 to 2
1. from 1 to 2

ans 1 = 3

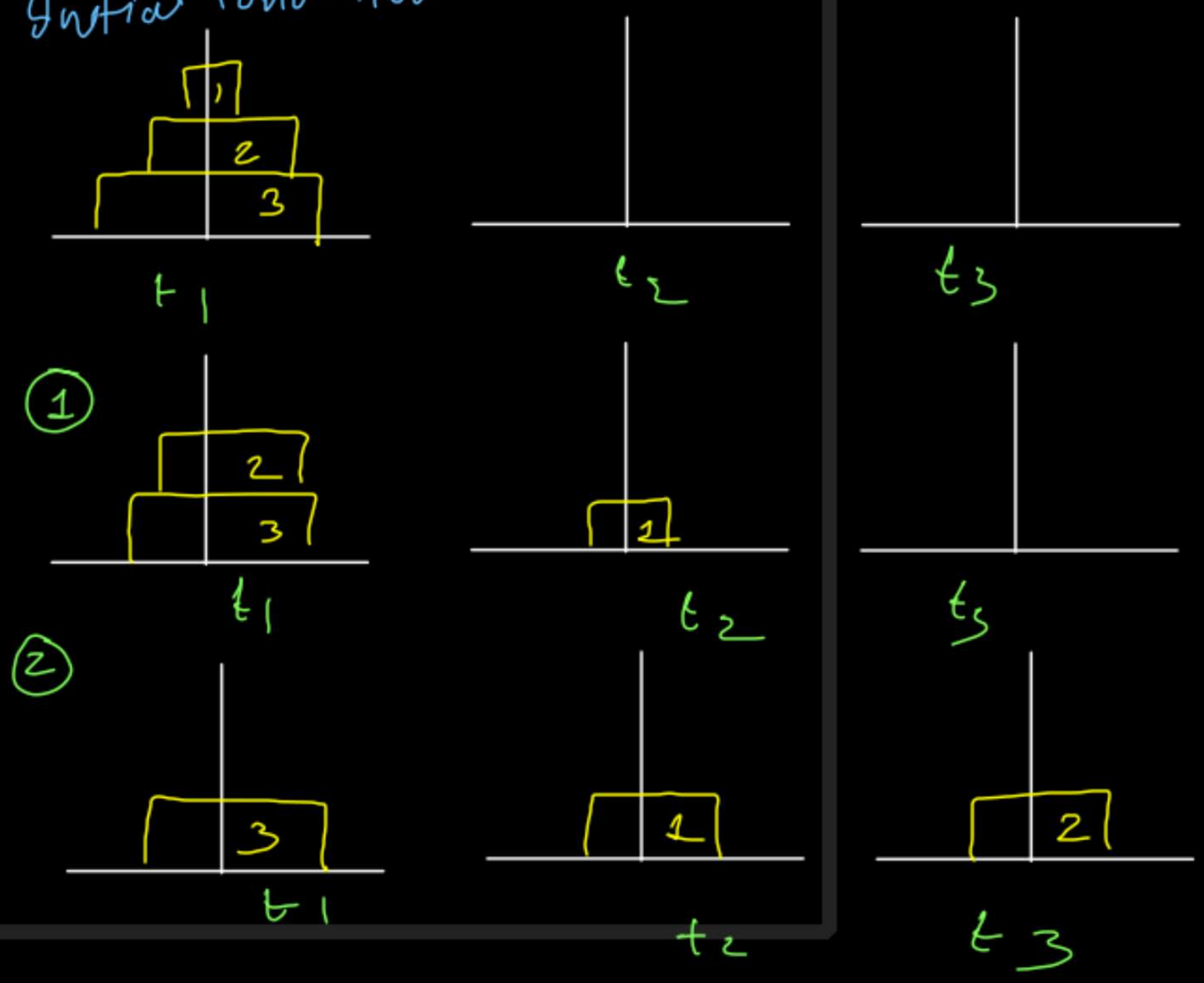
ans 2 = 3

+1 = 7

```java
// avoid space at the starting of the string in "move disk....."
class Hanoi {

    // t1 -> src t2-> dest t3-> aux
    public long toh(int N, int t1, int t2, int t3) {
        // Your code here

        if(N == 0) return 0;

        long ans1 = toh(N-1,t1,t3,t2);
        System.out.println("move disk " + N + " from rod " + t1 + " to rod " + t2);
        long ans2 = toh(N-1,t3,t2,t1);

        return ans1 + ans2 + 1;
    }
}
```

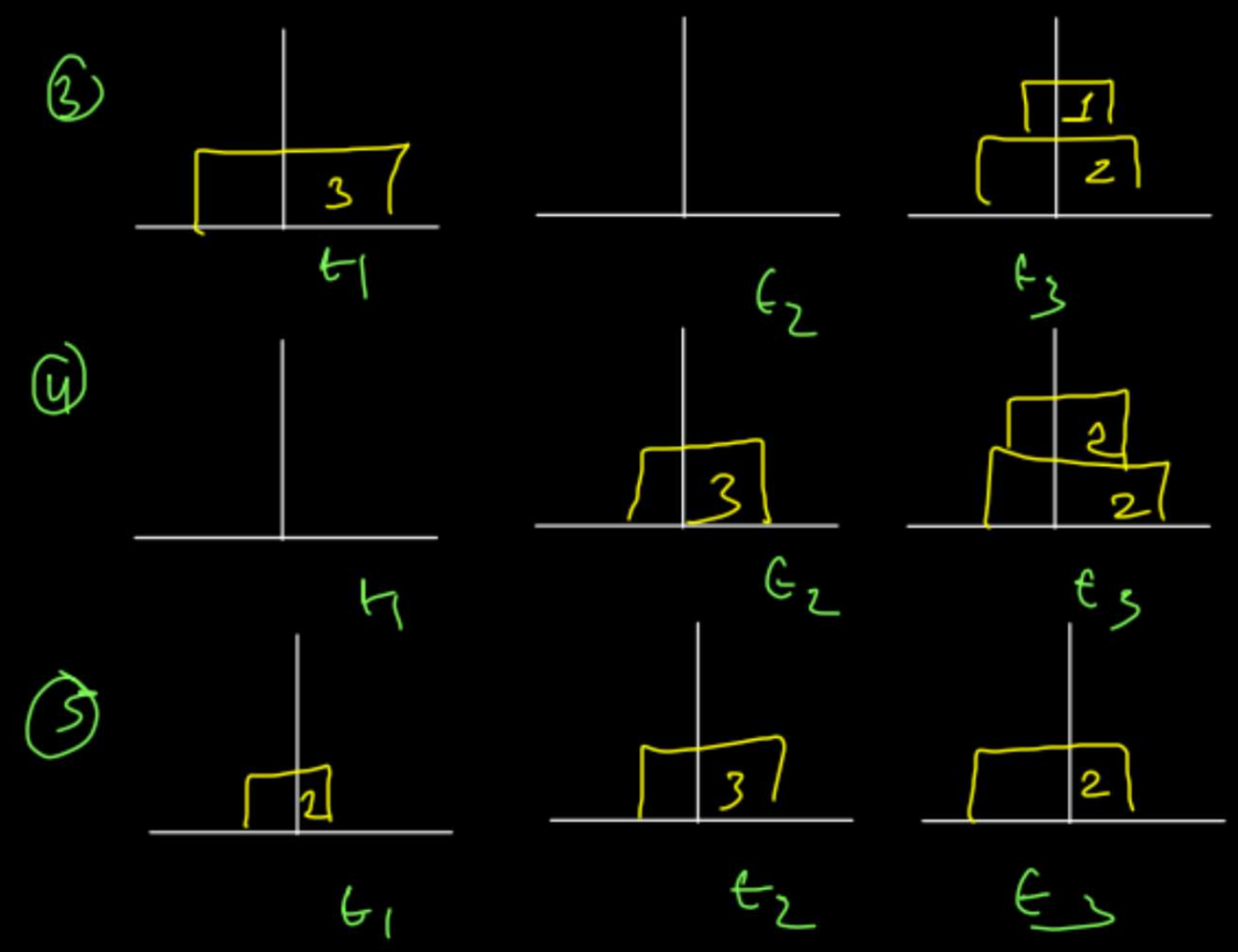
1. from 1 to 2
2. from 1 to 3
3. from 2 to 3
4. from 1 to 2
5. from 3 to 1
6. from 3 to 2
7. from 1 to 2

Initial condition

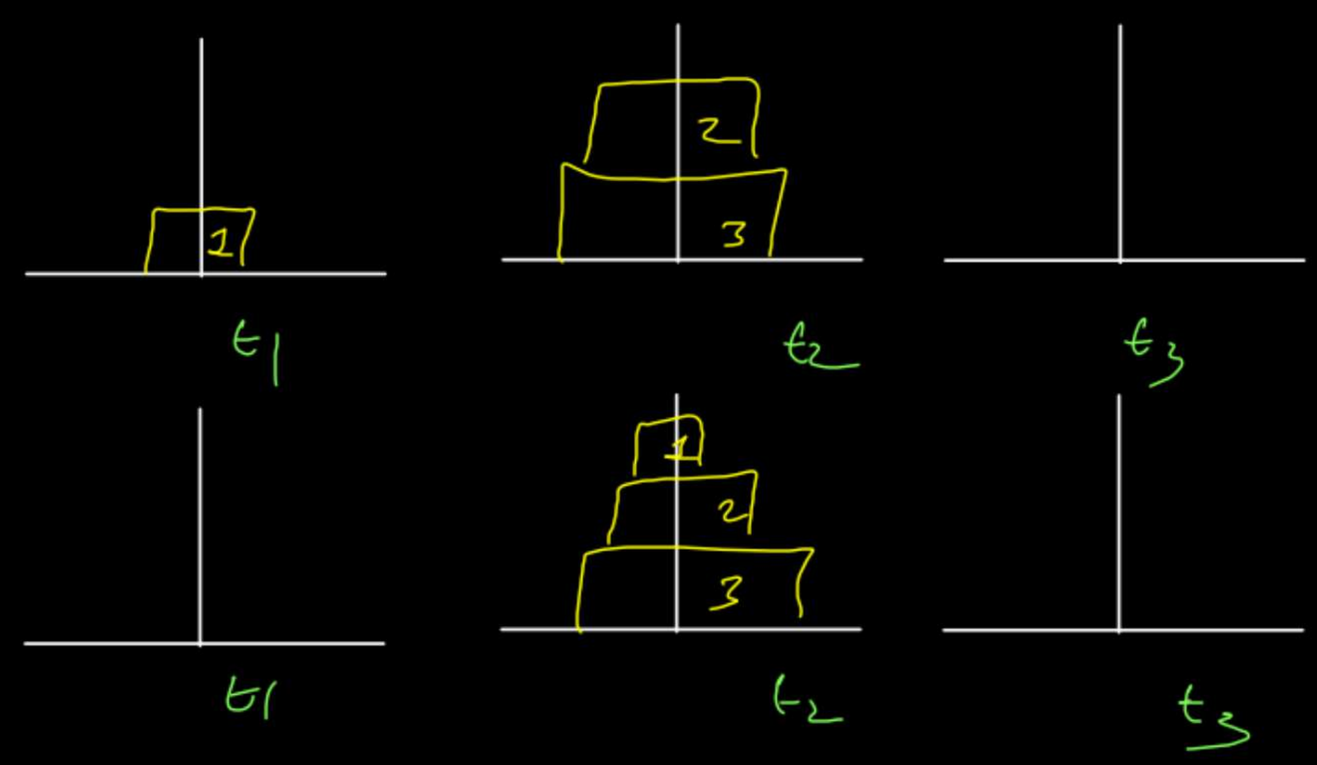


faith(4)

1. from 1 to 2
2. from 1 to 3
1. from 2 to 3
3. from 1 to 2
1. from 3 to 1
2. from 3 to 2
1. from 1 to 2



1. from 1 to 2
2. from 1 to 3
1. from 2 to 3
3. from 1 to 2
1. from 3 to 1
2. from 3 to 2
1. from 1 to 2



$t_1 = 1 \quad t_2 = 2 \quad t_3 = 3$

```java
// t1 -> src t2-> dest t3-> aux
public long toh(int N, int t1, int t2, int t3) {
    // Your code here

    if(N == 0) return 0;

    long ans1 = toh(N-1,t1,t3,t2); 1
    System.out.println("move disk " + N + " from rod " + t1 + " to ro
    long ans2 = toh(N-1,t3,t2,t1); 3

    return ans1 + ans2 + 1;  4
}
```

(1) from 1 to 2
(2) from 1 to 3
(1) from 2 to 3
(3) from 1 to 2
(1) from 3 to 1
(2) from 3 to 2
(1) from 1 to 2

src dest aux

$toh(N, t_1, t_2, t_3)$

$\ldots toh(3, t_1, t_2, t_3)$ → 7

$\ldots toh(2, t_1, t_3, t_2)$ $\quad$ $\ldots toh(2, t_3, t_2, t_1)$

$\ldots toh(1, t_1, t_2, t_3)$: $toh(1, t_2, t_3, t_1)$: $toh(1, t_3, t_1, t_2)$: $toh(1, t_1, t_2, t_3)$

$toh(0)$ $toh(0)$ $toh(0)$ $toh(0)$ $toh(0)$ $toh(0)$ $toh(0)$ $toh(0)$

Tree

(1) from 1 to 2
(2) from 1 to 3
(1) from 2 to 3
(3) from 1 to 2
(1) from 3 to 1
(2) from 3 to 2
(1) from 1 to 2

Stack

(1) from 1 to 2
(2) from 1 to 3
(1) from 2 to 3
(3) from 1 to 2
(1) from 3 to 1
(2) from 3 to 2
(1) from 1 to 2

Lecture - 21

→ Display Array

→ Reverse Array

→ Master formula Introduction (Preorder, Inorder & Postorder)

→ Power **

- → Linear 1
- → Linear 2
- → Logarithmic

Print Elements of Array

School Accuracy: **52.5%** Submissions: **31753** Points: **0**

Given an array **Arr** of size **N**, print all its elements.

Example 1:

Input:
N = 5
Arr[] = {1, 2, 3, 4, 5}
Output: 1 2 3 4 5

① Expectation:

0 1 2 3 4 5 6
{9, 8, 6, 3, 4, 7, 2}

display(arr, 0) → print space seperated elements. (N)
idx

9 8 6 3 4 7 2

② faith : display(arr, 1) → print space seperated elements (N-1)
idx+1

③ Meeting expec with faith :

idx idx

0 1 2 3 4 5 6
{9, 8, 6, 3, 4, 7, 2}

Print yourself + " "
↳ 9 _

display(arr, idx+1)
↳ 8 6 3 4 7 2

```java
void display(int[] arr, int idx) {

    |

    System.out.print(arr[idx] + " ");   1
    display(arr,idx + 1);               2
}
```

idx idx idx idx idx idx idx idx

0 1 2 3 4 5 6

{ 9, 8, 6, 3, 4, 7, 2 }

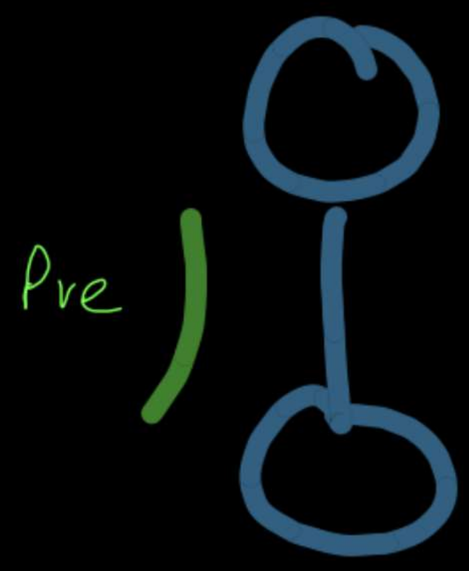
9_ 8_ 6_ 3_ 4_ 7_ 2_

```
if(idx = = arr.length)
    return;
```

→ Base Case

.. display (arr, 0)
↓
.. display (arr, 1)
↓
.. display (arr, 2)
↓
.. display (arr, 3)
↓
.. display (arr, 4)
↓
.. display (arr, 5)
↓
.. display (arr, 6)
↓
display (arr, 7)

```java
public static void displayArr(int[] arr, int idx){
    if(idx == arr.length) return;

    System.out.println(arr[idx]);
    displayArr(arr,idx + 1);
}
```



Reverse an Array

School Accuracy: **40.0%** Submissions: **100k+** Points: **0**

Given an array **A** of size **N**, print the reverse of it.

Example:

Input:
1
4
1 2 3 4
Output:
4 3 2 1

Expectation :

lo hi

0 1 2 3 4

{ 1, 2, 3, 4, 5 }

↓

lo hi

0 1 2 3 4

{ 5, 4, 3, 2, 1 }

reverse (arr, lo, hi);

lo → 0, hi → arr.length - 1

faith :

lo hi

0 1 2 3 4

{ 1, 2, 3, 4, 5 }

↓

lo hi

0 1 2 3 4

{ 1, 4, 3, 2, 5 }

reverse(arr, lo+1, hi-1);

Meeting expectation with faith :

① Swap lo & hi

② reverse (arr, lo+1, hi-1)

lo hi

0 1 2 3 4

{ 5, 2, 3, 4, 1 }

s l

① 0 1 2 3 4

{ 5, 2, 3, 4, 1 }

lo hi

② 0 1 2 3 4

{ 5, 2, 3, 4, 1 } ⟶ 0 1 2 3 4 { 5, 4, 3, 2, 1 }

```java
public static void reverse(int[] arr, int lo, int hi) {

    //swap yourself
    int temp = arr[lo];
    arr[lo] = arr[hi];
    arr[hi] = temp;

    reverse(arr,lo+1,hi-1);
}
```

① ②

lo lo lo hi hi hi

0 1 2 3 4

{1, 2, 3, 4, 5}

5 4 2 1

..reverse(arr, 0, 4)

↓

..reverse(arr, 1, 3)

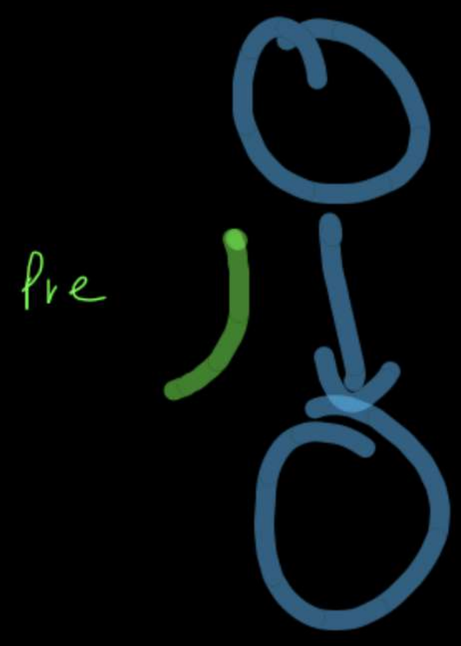
↓

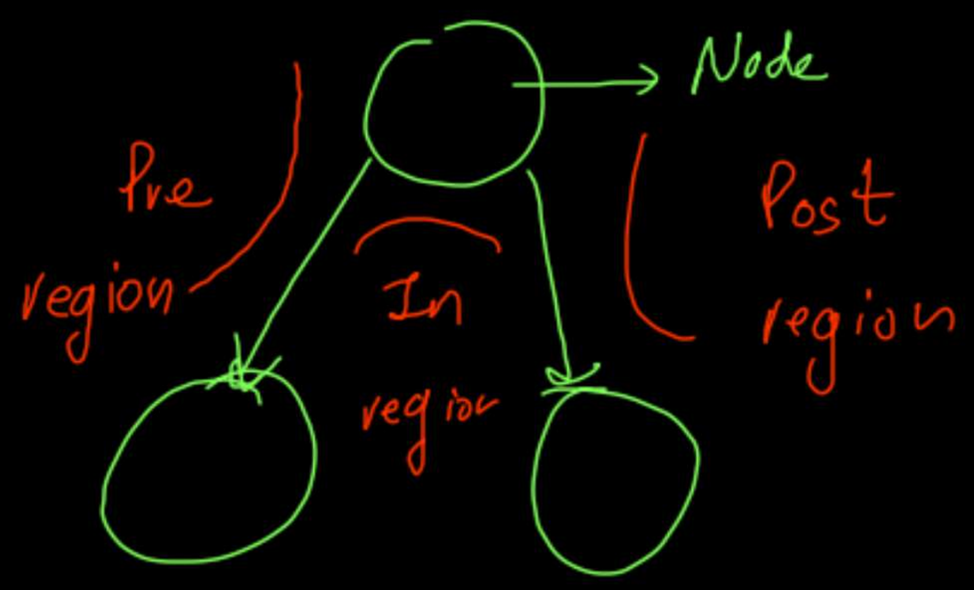
reverse(arr, 2, 2)

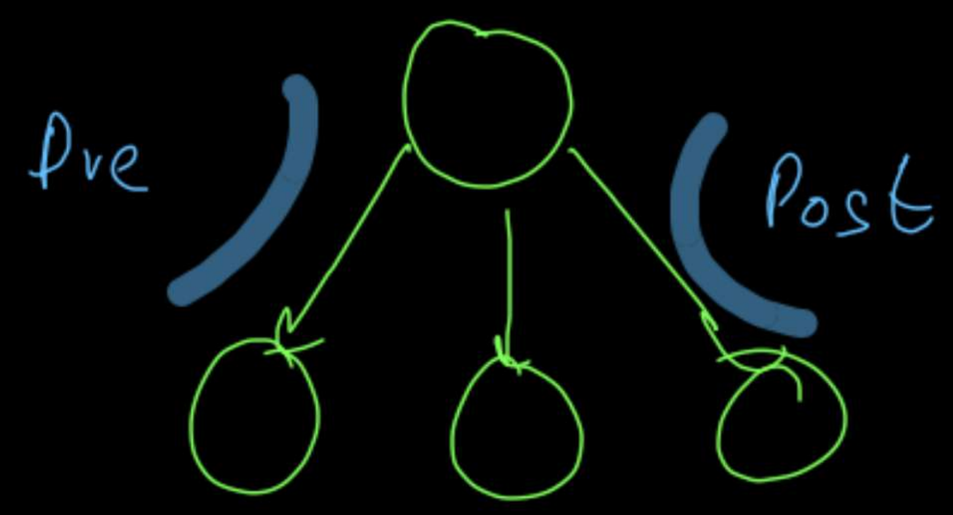
if (lo >= hi) return;

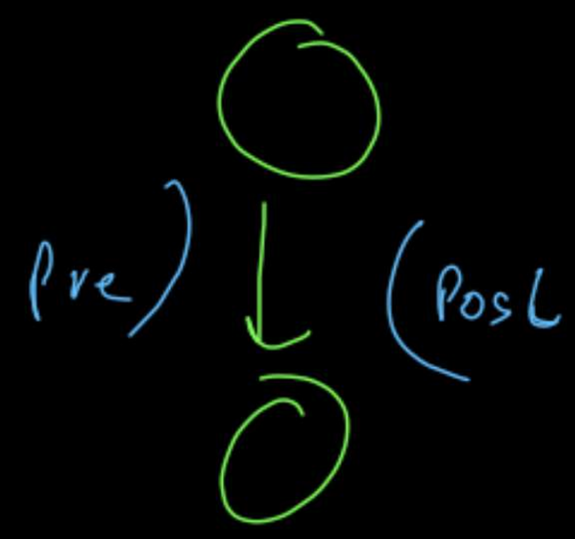
Base case

```java
public static void reverse(int[] arr, int lo, int hi) {

    if(lo >= hi) return;

    //swap yourself
    int temp = arr[lo];
    arr[lo] = arr[hi];
    arr[hi] = temp;

    reverse(arr,lo+1,hi-1);
}
```




Node
Pre
region
In
region
Post
region


Pre
Post


Pre
Post

Recursion Master formula

$$\text{Time Complexity} = (\text{calls})^{\text{height}} + (\text{Pre} + \text{Post}) * \text{height}$$

This is max no of calls from a node in the tree.

50. Pow(x, n)

Medium 5801 6346 Add to List Share

Implement pow(x, n), which calculates `x` raised to the power `n` (i.e., x^n).

Example 1:

```
Input: x = 2.00000, n = 10
Output: 1024.00000
```

① Expectation : power(x, n) → $\underbrace{x * x * x \cdots x}_{n \text{ times}}$

power(2, 7) → $2 * 2 * 2 * 2 * 2 * 2 * 2$

② faith : power(x, n-1) → $\underbrace{x * x * x \cdots x}_{(n-1) \text{ times}}$

power(2, 6) → $2 * 2 * 2 * 2 * 2 * 2$

③ Meeting expectation with faith : power(x, n) = power(x, n-1) * x

$$x^n = x * x^{n-1}$$

x^N

$\downarrow$

x^{N-1}

$\downarrow$

x^{N-2}

$\vdots$

$(num)^0 \rightarrow 1$

$0 \longrightarrow$ if (n == 0) return 1

$$x^{-N} = \frac{1}{x^N}$$

$$2^{-6} = \frac{1}{2^6}$$

$$2^{-6} = \frac{1}{2^{-(-6)}}$$

```java
class Solution {

    public double power(double x, int n) {
        if(n == 0) return 1;

        return x * power(x,n-1);

    }

    public double myPow(double x, int n) {
        if(n < 0) {
            return 1.0 / power(x,-n);
        }

        return power(x,n);
    }
}
```

This gives stack overflow

```java
class Solution {

    public double power(double x, int n) {
        if(n == 0) return 1;

        return x * power(x,n-1);

    }

    public double myPow(double x, int n) {
        if(n < 0) {
            return 1.0 / power(x,-n);
        }

        return power(x,n);
    }
}
```

pow(2,5) ↑ 2*16 = 32

↓

pow(2,4) ↑ 2*8 = 16

↓

pow(2,3) ↑ 2*4 = 8

↓

pow(2,2) ↑ 2*2 = 4

↓

pow(2,1) ↑ 2*1 = 2

↓

pow(2,0) ↑ 1

$(\text{calls})^{h} + (\text{pre} + \text{post}) * N$

$(1)^{N} + (k1 + k2) * N = 1 + k^{N} \approx O(N)$

power(2,6) → power(2,3) * power(2,3)

power(2,7) → 2* pow(2,3) * pow(2,3)

x^N

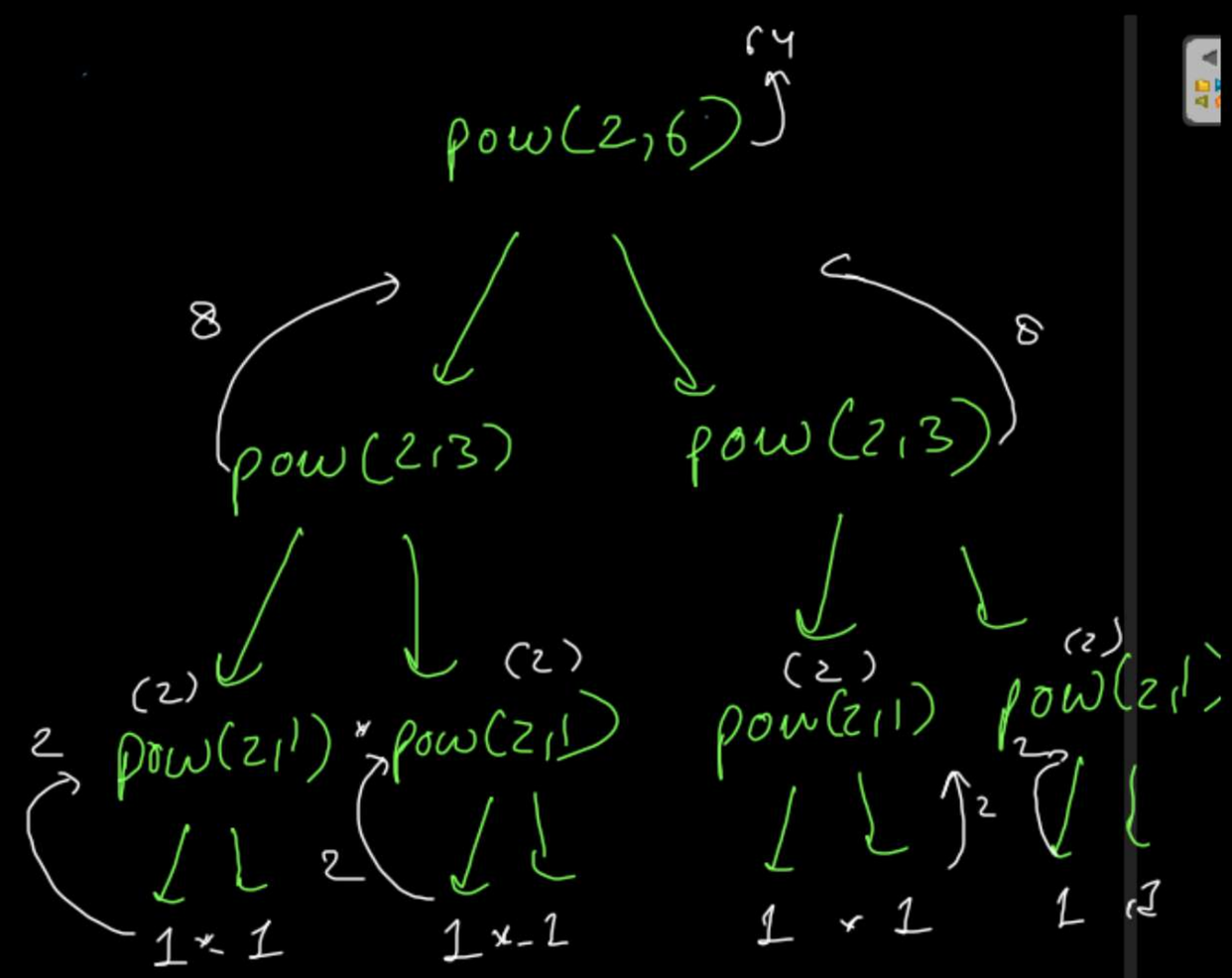
if (N % 2 == 0)

pow(x,n/2) * pow(x,n/2)

else

x * pow(x,n/2) * pow(x,n/2)

```java
class Solution {

    public double power(double x, int n) {
        if(n == 0) return 1;

        if(n % 2 == 0)
            return power(x,n/2) * power(x,n/2);

        return x * power(x,n/2) * power(x,n/2);

    }

    public double myPow(double x, int n) {
        if(n < 0) {
            return 1.0 / power(x,-n);
        }

        return power(x,n);
    }
}
```



(calls)^h + (pre+post) * h

$(2)^h$ + (k) * h =

$(2)^{\log_2 N}$ + (k) * $\log_2 N$

N + $\log_2 N$ = O(N)

pow(8)

↓

4

↓

2

↓

1

4

$\log_2 8 + 1$

$\approx O(\log_2 N)$

```
public double power(double x, int n) {
    if(n == 0) return 1;

    double smallAns = power(x,n/2);

    if(n % 2 == 0)
        return smallAns * smallAns;

    return x * smallAns * smallAns;

}

public double myPow(double x, int n) {
    if(n < 0) {
        return 1.0 / power(x,-n);
    }

    return power(x,n);
}
```

pow(2,6)

↓

pow(2,3)

↓

pow(2,1)

↓

a

$(\text{calls})^h + (\text{pre} + \text{post}) * h$

$(1)^{\log_2 N} + K * \log_2 N$

$1 + \log_2 N \approx O(\log_2 N)$