PW SKILLS

Recursion

What and Why?

If you don't study recursion → Trees, DP, Graph
↓
DFS

Recursion is not a D.S., it is an algo
↓
We believe in Recursion

Recurrence Relations
↓
Big Problem → using Smallerer subproblem

Function calls

Methods → return statement
return type
parameters

```
public class FunctionCalls {
    public static void apple(){
        System.out.println("Hi, I am in apple");
    }
    public static void main(String[] args) {
        System.out.println("Hi, I am in main method");
        apple();
    }
}
```

- Hi, I am in main
- Hi, I am in apple

Function calls

```
public static void mango(){
    System.out.println("Hi, I am in mango");
}
public static void banana(){
    System.out.println("Hi, I am in banana");
    mango();
}
public static void apple(){
    System.out.println("Hi, I am in apple");
    banana();
}
public static void main(String[] args) {
    System.out.println("Hi, I am in main");
    apple();
}
```

Output

- Hi, I am in main
- Hi, I am in Apple
- Hi, I am in Banana
- Hi, I am in Mango

Function calls

```
public static void mango(){
  5 System.out.println("Hi, I am in mango");8
}
public static void banana(){
  4 mango();
  5 System.out.println("Hi, I am in banana");
}
public static void apple(){
  3 banana();
  6 System.out.println("Hi, I am in apple");
  7 mango();
}
public static void main(String[] args) {
  1 System.out.println("Hi, I am in main");
  2 apple();
}
```

Factorial Recurrence Relation

PW SKILLS

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

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

$6! = 6 \times 5!$

$8! = 8 \times 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1$

$8! = 8 \times 7!$

$n! = n \times n-1 \times n-2 \times n-3 \times \cdots 3 \times 2 \times 1$

$n! = n \times (n-1)!$

$f(n) = n * f(n-1)$

$f(x) = x!$

$f(3) = 3!$, $f(2) = 2!$

Ques:

Q1 : Make a function which calculates the factorial of n using recursion.

```
int fact(int n){
    if(n==1) return 1;
    ans = n* fact(n-1);
    return ans;
}
```

$$fact(n) = n * fact(n-1)$$

Ques:

Q1 : Make a function which calculates using recursion.

```
public static int fact(int n){
    int ans = n * fact(n-1);
    return ans;
}
```

Base Case

```
public static int fact(int n){
    int ans = 1 * fact(n-1);
    return ans;
}
```

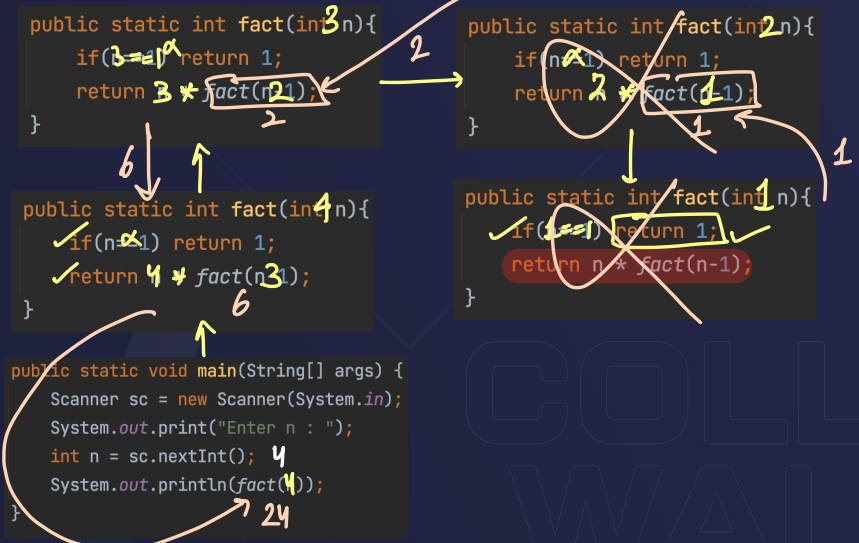
```
public static int fact(int n){
    int ans = 2 * fact(n-1);
    return ans;
}
```

```
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.print("Enter n : ");
    int n = sc.nextInt(); → 3
    System.out.println(fact(3));
}
```

```
public static int fact(int n){
    int ans = n * fact(n-1);
    return ans;
}
```

aaaaa not reqd.

Dry Run :


public static int fact(int n){
if(n==1) return 1;
return n * fact(n-1);
}
public static int fact(int n){
if(n==1) return 1;
return n * fact(n-1);
}
public static int fact(int n){
if(n==1) return 1;
return n * fact(n-1);
}
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
System.out.print("Enter n : ");
int n = sc.nextInt();
System.out.println(fact(n));
}

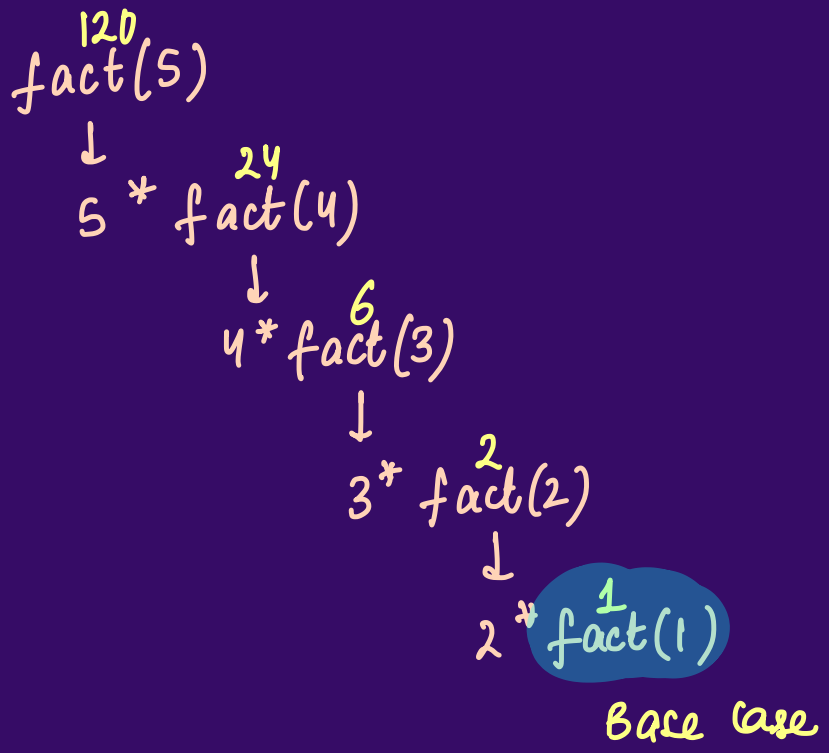
Ques:

11 = 1

PW SKILLS

Q1 : Make a function which calculates the factorial of n using recursion.

n = 5



Ques:

PW SKILLS

Q2 : Print n to 1 → using Recursion

take a number 'n' as input & print n to 1.

For Ex : n = 5 Output : 5
4
3
2
1

n = 3 Output : 3
2
1

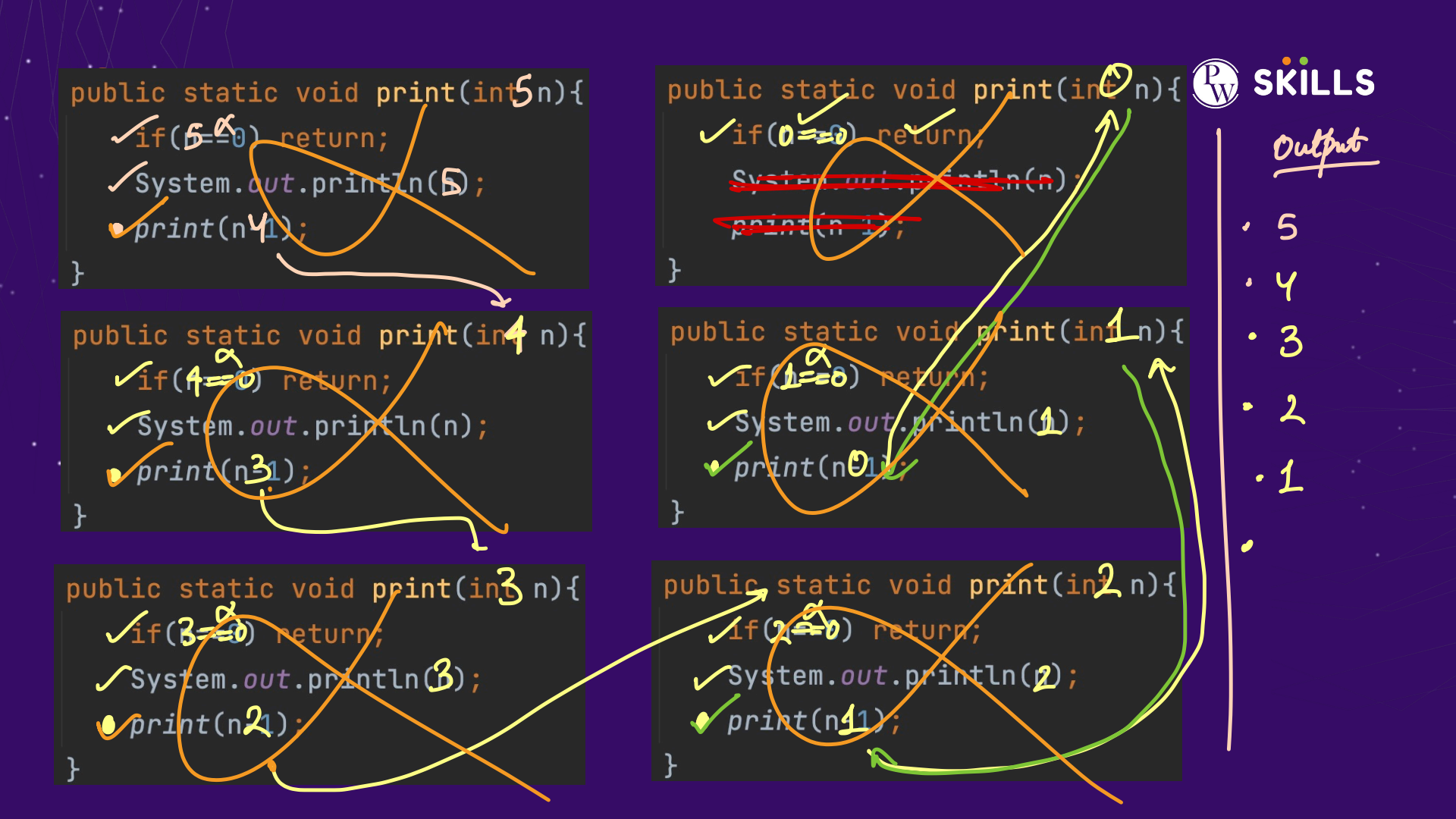
Ques:

Q2 : Print n to 1

print(5)
↓
5
print(4)

print(4)
↓
4
print(3)

```java
public static void print(int n){
    if(n==0) return; // base case
    System.out.println(n); // work
    print(n-1); // call
}
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.print("Enter n : ");
    int n = sc.nextInt();
    print(n);
}
```


public static void print(int n){
if(n==0) return;
System.out.println(n);
print(n-1);
}
public static void print(int n){
if(0==0) return;
System.out.println(n);
print(n-1);
}
public static void print(int n){
if(1==0) return;
System.out.println(n);
print(n-1);
}
public static void print(int n){
if(2==0) return;
System.out.println(n);
print(n-1);
}
public static void print(int n){
if(3==0) return;
System.out.println(n);
print(n-1);
}
public static void print(int n){
if(4==0) return;
System.out.println(n);
print(n-1);
}
SKILLS
Output
5
4
3
2
1

Ques:

PW SKILLS

Q3 : Print 1 to n (extra parameter)

Take 'n' as input & print 1 to n

Ex = n = 4 → Output =

1
2
3
4

Ques:

Q3 : Print 1 to n (extra parameter)

Global variable

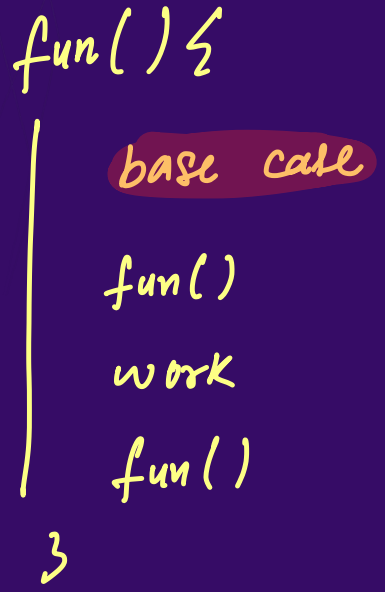
```java
static int n;
public static void print(int x){
    if(x>n) return; // base case
    System.out.println(x); // work
    print(x+1); // call
}
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.print("Enter n : ");
    n = sc.nextInt();
    print(1);
}
```

Extra Parameter

```java
public static void print(int x, int n){
    if(x>n) return; // base case
    System.out.println(x); // work
    print(x+1,n); // call
}
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.print("Enter n : ");
    int n = sc.nextInt();
    print(1,n);
}
```

Ques:

Q4 : Print 1 to n (after recursive call)

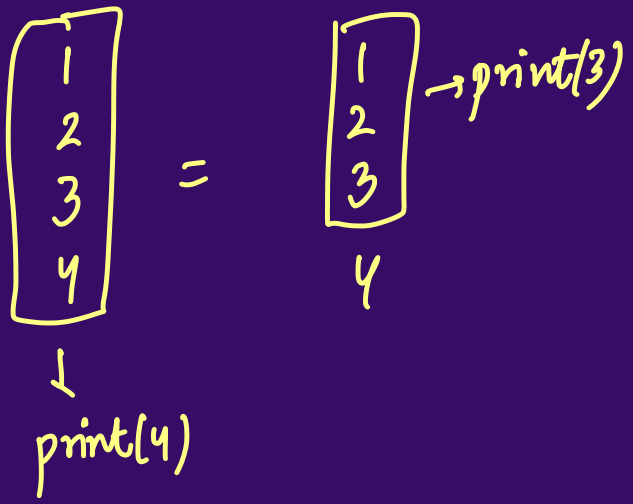


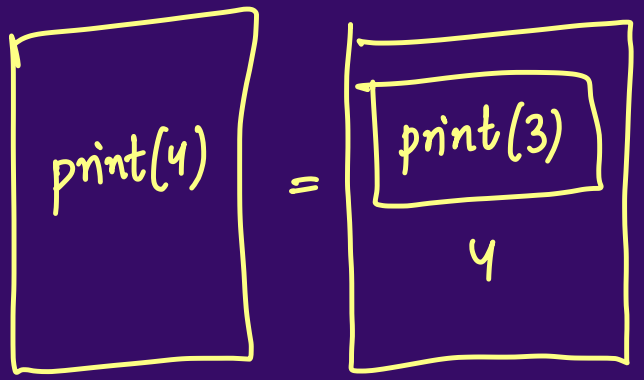

```java
public static void print(int n){
    if(n==0) return; // base case
    print(n-1); // call
    System.out.println(n); // work
}
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.print("Enter n : ");
    int n = sc.nextInt();
    print(n);
}
```

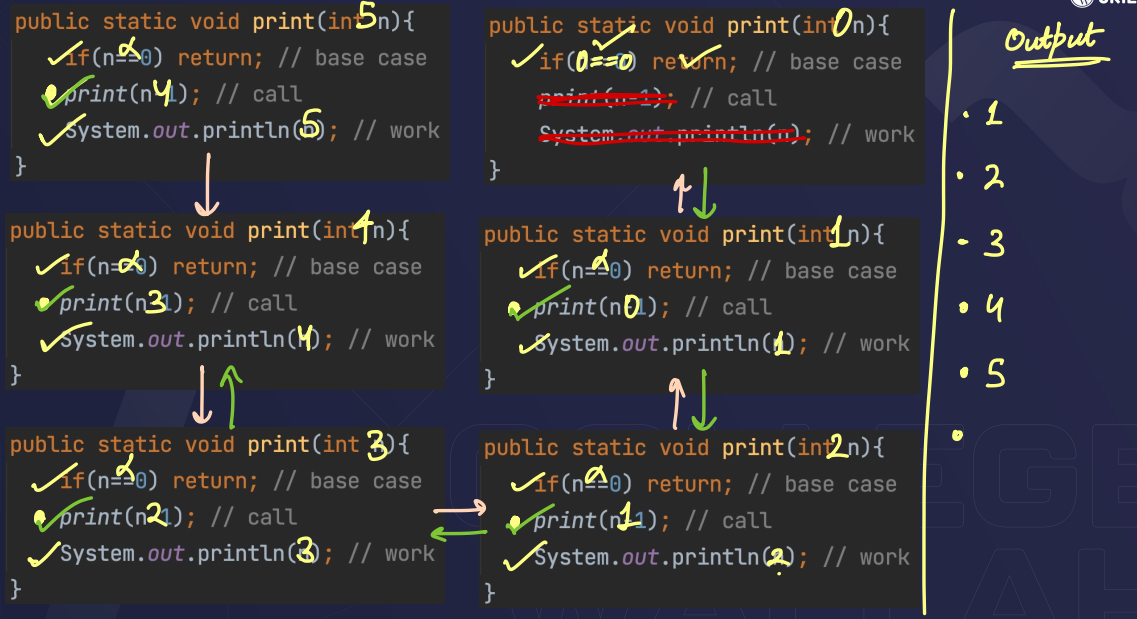
Ques: print(n) will output 1 2 ⋮ n

Q4 : Print 1 to n (after recursive call)

```
public static void print(int n){
    if(n==0) return; // base case
    print(n-1); // call
    System.out.println(n); // work
}
```






public static void print(int n){
if(n==0) return; // base case
print(n-1); // call
System.out.println(n); // work
}
Output
1
2
3
4
5

Ques:

PW SKILLS

Q5 : Print sum from 1 to n (Parameterised)

→ Recursion

$n = 5 \rightarrow Sum = 1+2+3+4+5 = 15 \rightarrow ans$

or

$5+4+3+2+1$

```
Sum = 0;
for(int i=1; i<=n; i++){
    Sum += i;
}
```

3

Iterative

n ↑ Calculate sum ↑

Sum(5, 0)
↓
Sum(4, 5)
↓
Sum(3, 9)
↓
Sum(2, 12) → Sum(1, 14)

base case ↑

Sum(0, 15) ↑

Ques:

Q5 : Print sum from 1 to n (Parameterised)

```java
public static void sum(int n, int s){
    if(n==0){ // base case
        System.out.println(s);
        return;
    }
    sum(n-1,s+n); // call and work
}
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.print("Enter n : ");
    int n = sc.nextInt();
    sum(n,0);
}
```

Printing the sum from 1 to n or n to 1

Ques:

PW SKILLS

Q6 : Print sum from 1 to n (Return type)

↓

Recurrence Relation

$$Sum(n) = n + \underbrace{n-1 + n-2 + \cdots 3 + 2 + 1}$$

$$Sum(n) = n + Sum(n-1)$$

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

Ques:

PW SKILLS

Q7 : Make a function which calculates 'a' raised to the power 'b' using recursion. Take 'a' & 'b' input from user.

pow(a, b)

↓

calculate a^b

$$a^b = \underbrace{a \times a \times a \times \ldots \; a}_{b \text{ times}}$$

$$a^b = \underbrace{a}_{1} \times \underbrace{a \times a \times a \ldots \ldots a}_{b-1 \text{ times}}$$

$$a^b = a \times a^{b-1}$$

$$pow(a,b) = a * pow(a, b-1)$$

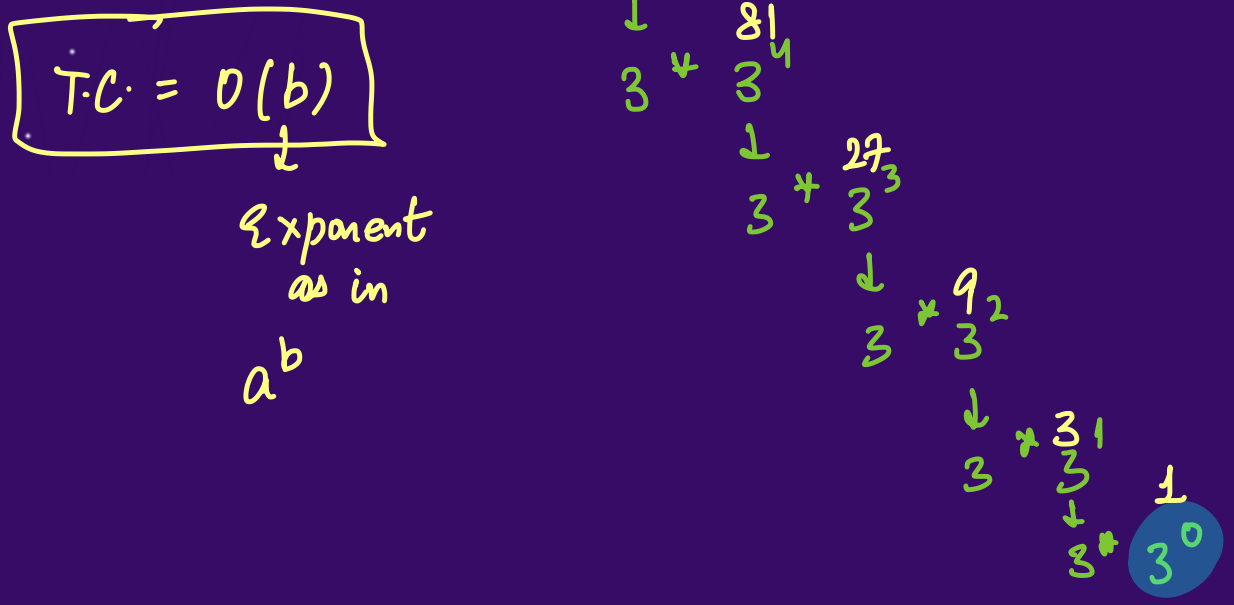
Hint

$$x^m \cdot x^n = x^{m+n}$$

Ques:

Q7 : Make a function which calculates ‘a’ raised to the power ‘b’ using recursion.

```
if(b==0) return 1;
return a * pow(a,b-1);
```



Ques:

#Hint : $x^{m+n} = x^m \cdot x^n$

SKILLS

Q8 : Power function (logarithmic) $a^b \rightarrow$ T.C. $= O(\log_2 b)$

$2^{64} = 2 * 2^{63}$

$2^{63} = 2 * 2^{62}$

$2^{62} = 2 * 2^{61}$

.

.

.

.

.

$2^1 = 2 * 2^0$

very slow

64 calls (bekar & slow)

$2^{64} = 2^{32} * 2^{32}$

$2^{32} = 2^{16} * 2^{16}$

$2^{16} = 2^8 * 2^8$

$2^8 = 2^4 * 2^4$

$2^4 = 2^2 * 2^2$

$2^2 = 2^1 * 2^1$

↓

6 calls $= \log_2 64$

$a^b = (a^{b/2})^2$

Ques:

PW SKILLS

Q8 : Power function (logarithmic)

$a^b = a^{b/2} * a^{b/2}$

int ans = pow(a, b/2);

pow(a, b) = ans * ans;

↓

Wrong

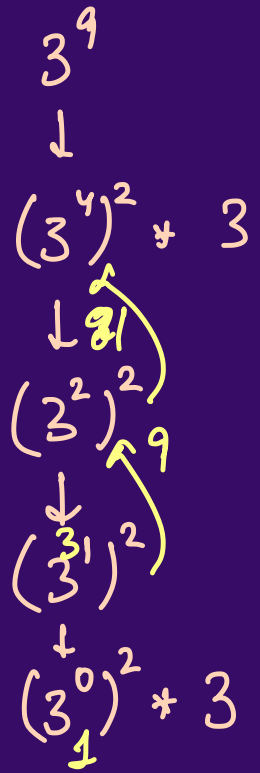
$2^5 = 2^2 * 2^2 * 2$ → extra

$3^7 = 3^3 * 3^3 * 3$

$3^8 = 3^4 * 3^4$

Ques:

Q8 : Power function (logarithmic)



*Multiple Calls

PW SKILLS

Q9 : Write a function to calculate the nth fibonacci number using recursion.

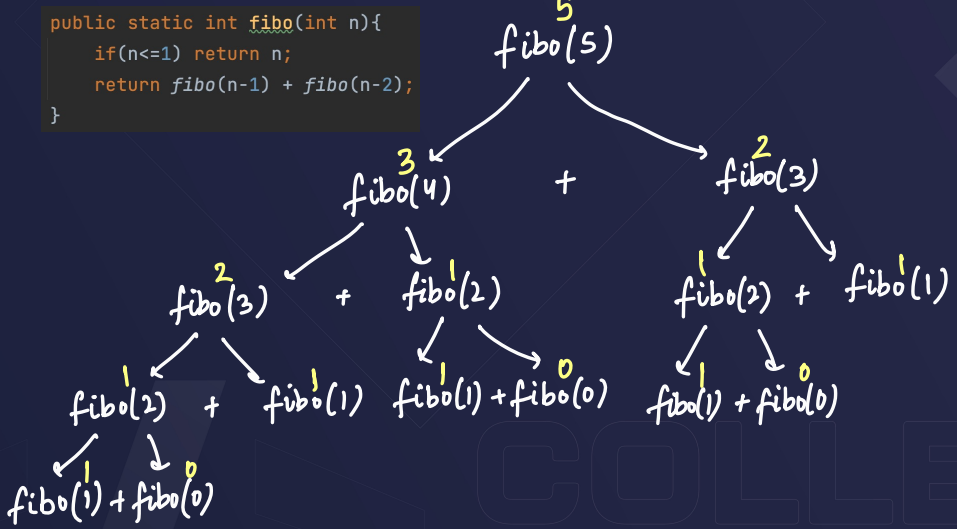
n =	0	1	2	3	4	5	6	7	8	9	10	11
	0	1	1	2	3	5	8	13	21	34	55	89 . . .

$$fibo(n) = fibo(n-1) + fibo(n-2)$$

base case ⇒ `if( n<=1 ) return n;`

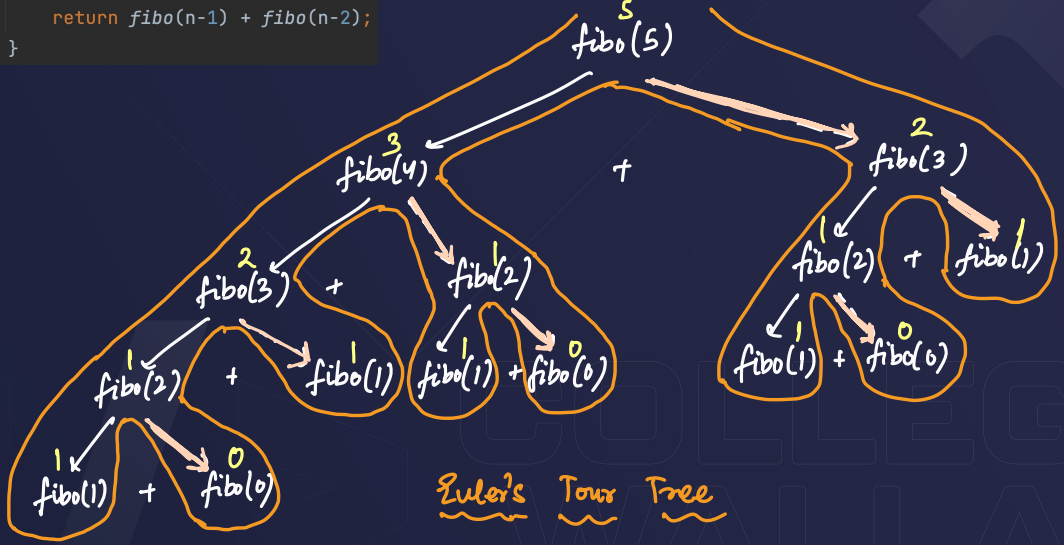
[Leetcode 509]

```java
public static int fibo(int n){
    if(n<=1) return n;
    return fibo(n-1) + fibo(n-2);
}
```



Recursive Tree

```
public static int fibo(int n){
    if(n<=1) return n;
    return fibo(n-1) + fibo(n-2);
}
```

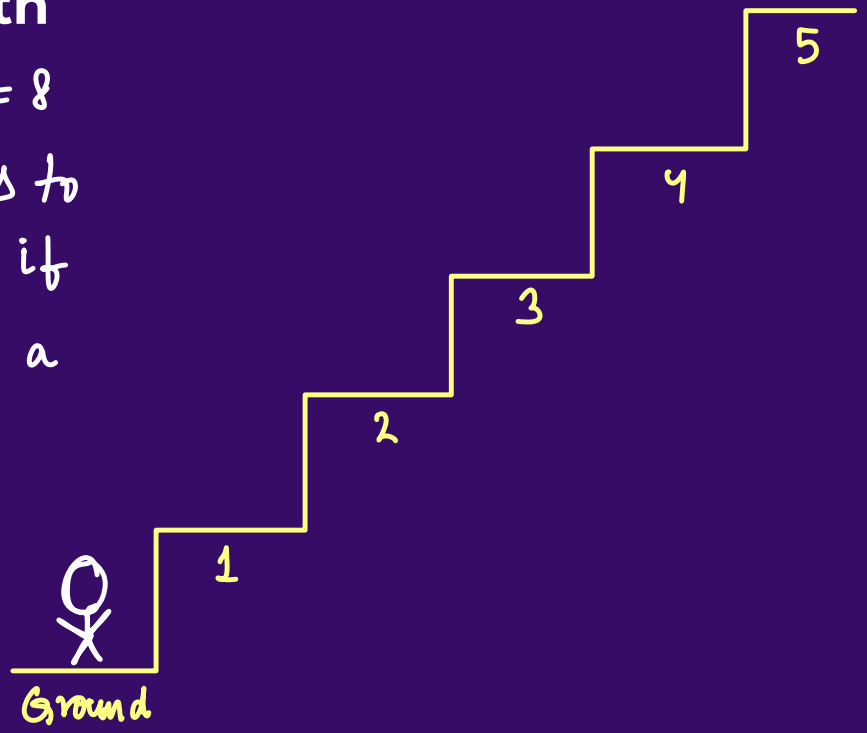


Ques:

Q3 : Stair Path

$n = 5 \rightarrow$ ways = 8

Find no. of ways to reach n^{th} stair if 1 or 2 jump at a time is allowed

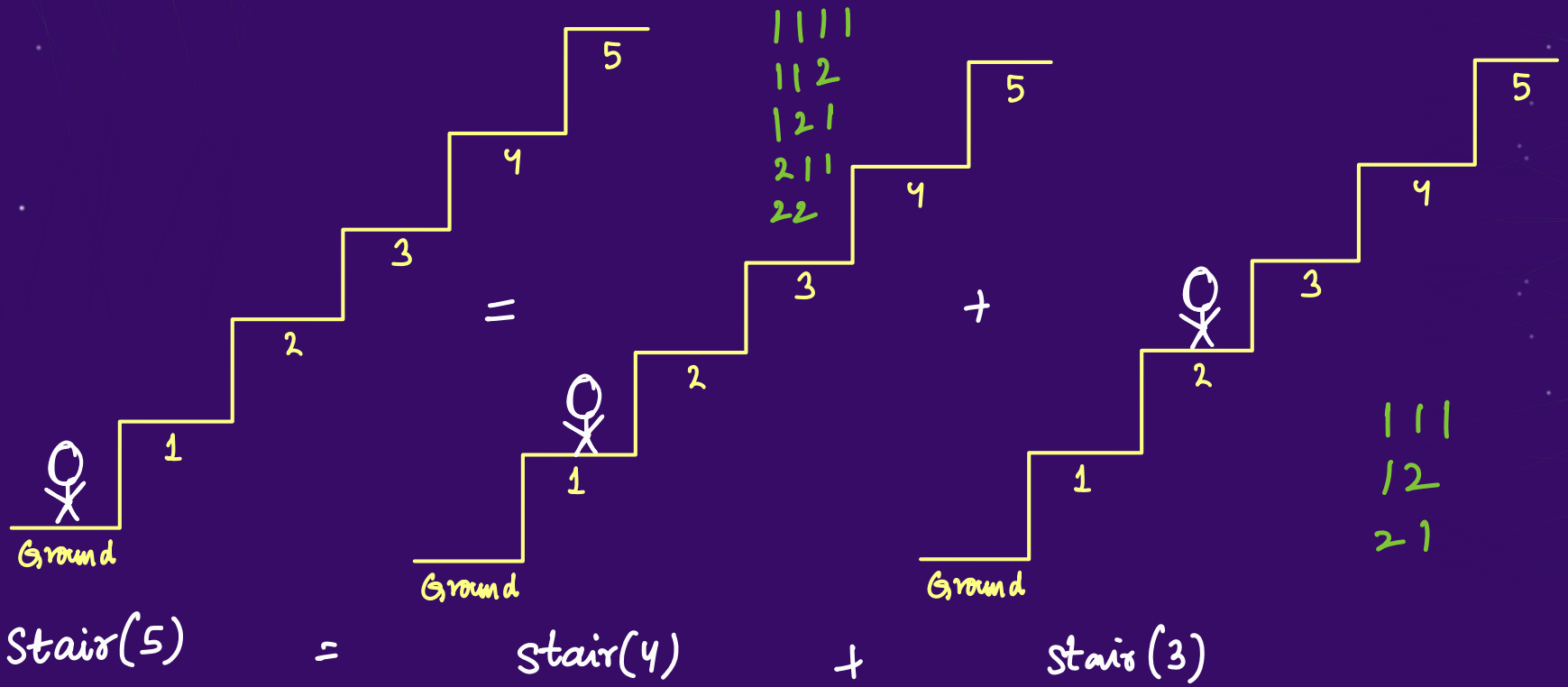


Ways :

1 1 1 1 1
1 1 1 2
1 1 2 1
1 2 1 1
2 1 1 1
2 1 2
2 2 1
1 2 2

Ques:

Q3 : Stair Path

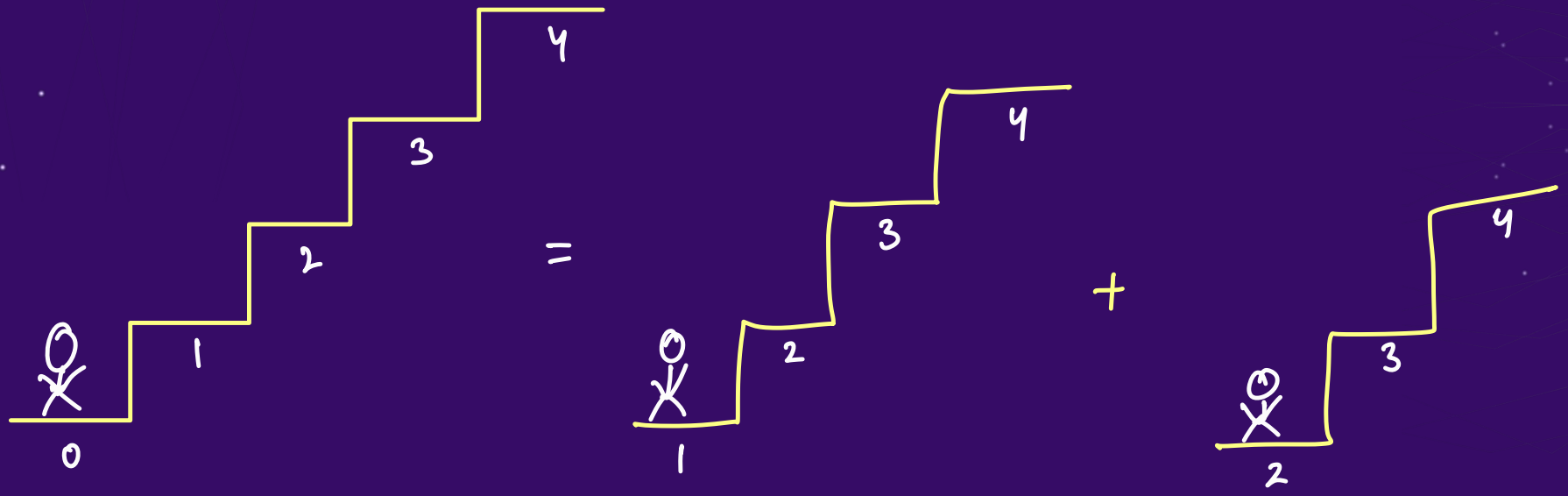


Ques:

Q3 : Stair Path

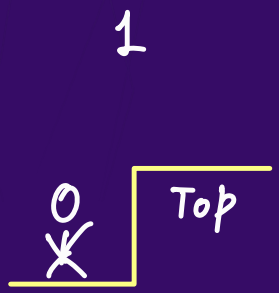
$$Stair(n) = Stair(n-1) + Stair(n-2)$$

Exactly same like fibonacci

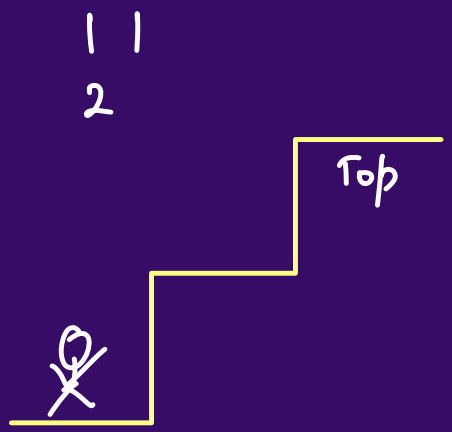


Ques:

Q3 : Stair Path



Stair (1) = 1



Stair (2) = 2

Homework

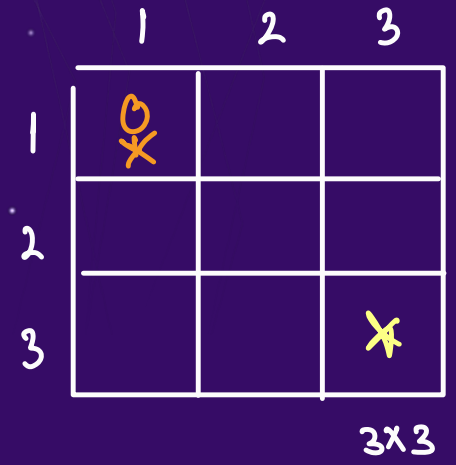
n^{th} stair (1 or 3 jumps)

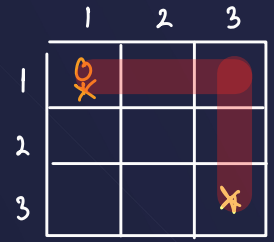
Ques:

Rat in a maze

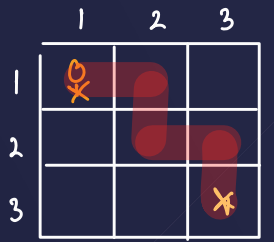
Q4 : Maze path [Very Important]

Given a m×n grid
you have to reach from top left corner to bottom right corner.
you can go either down or right at a time. Find the no. of paths.



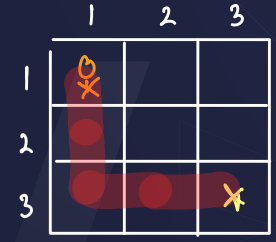


RRDD

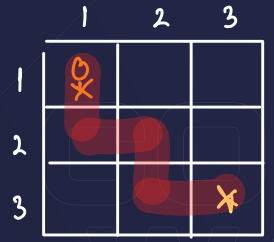


RDRD

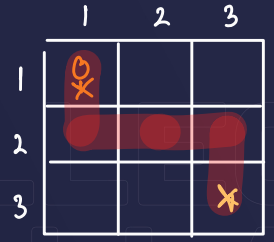
RDDR



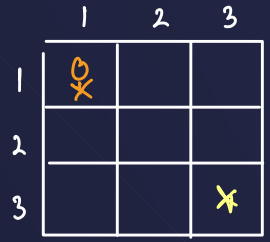
DDRR



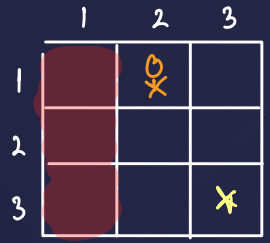
DRDR



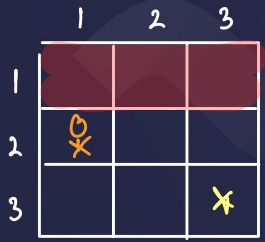
DRRD



=



+



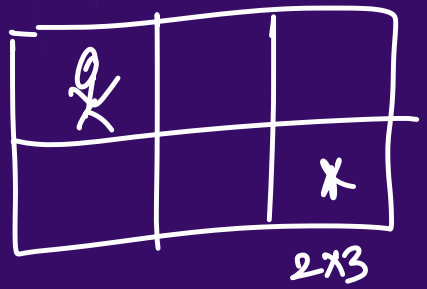
$(1,1) \rightarrow (3,3)$ = $(1,2) \rightarrow (3,3)$ + $(2,1) \rightarrow (3,3)$

total ways = rightways + downways

Base Case → if (row == m or col == n) return 1;

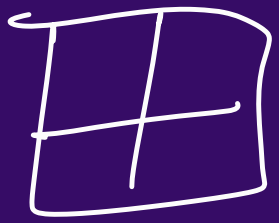
Ques:

Q4 : Maze path



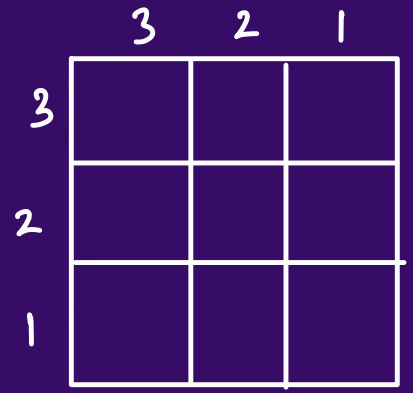
RRD
RDR
DRR

(3)



Ques:

Q4 : Maze path



(m, n) → (1, 1)

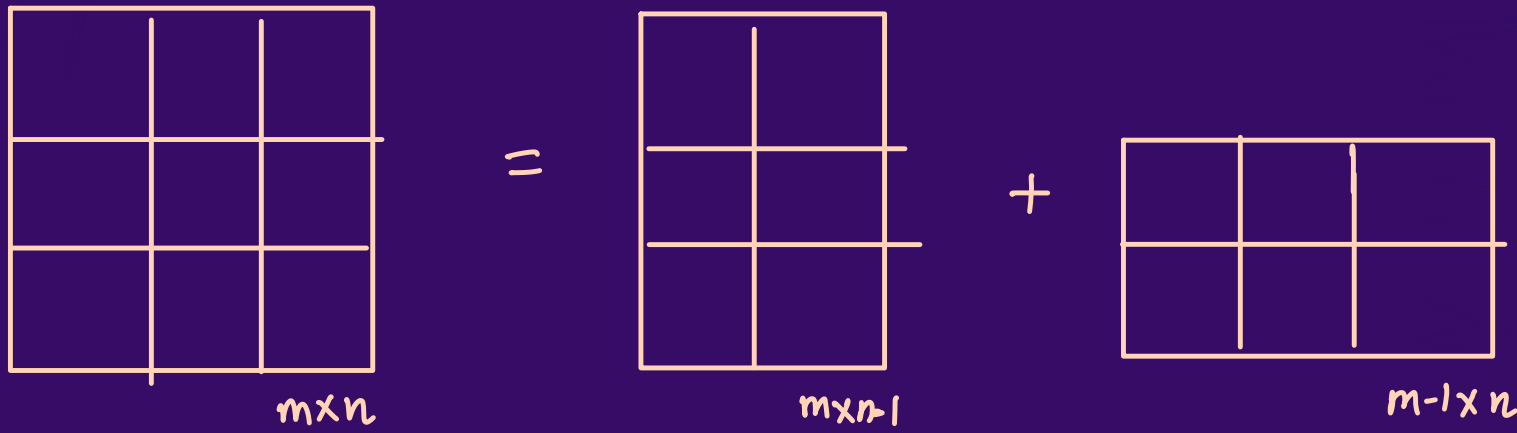
or

(1, 1) → (m, n)

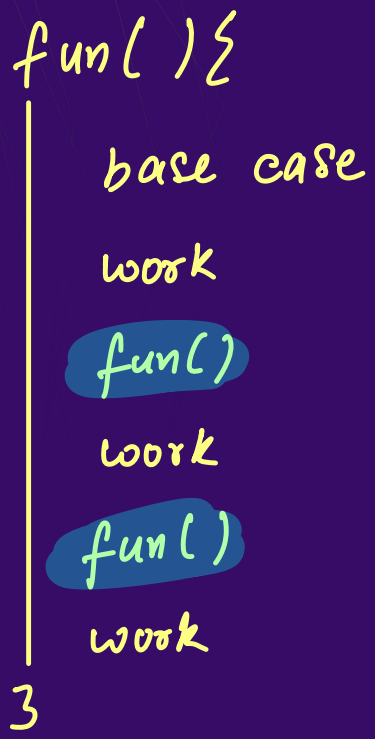
right → col--

down → row--

```java
public static int maze2(int m, int n){
    if(m==1 || n==1) return 1;
    int rightWays = maze2(m,n-1);
    int downWays = maze2(m-1,n);
    return rightWays + downWays;
}
```



Pre In Post

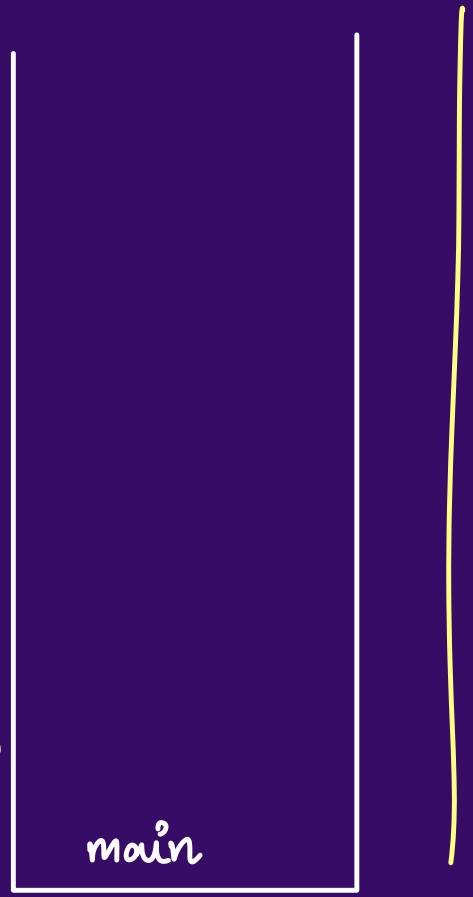


```java
public static void pip(int n){
  if(n==0) return;
  System.out.println(n);
  pip(n-1);
  System.out.println(n);
  pip(n-1);
  System.out.println(n);
}
```

pip(3)

3 2 1 1 1 2 1 1 1 2 3 2 1 1 1 2 1 1 1 2 3

Call Stack



Call Stack

```java
public static void pip(int n){
    if(n==0) return;
    System.out.println(n);
    pip(n-1);
    System.out.println(n);
    pip(n-1);
    System.out.println(n);
}
```

pip(3)

3 2 1 1 1 2 1 1 1 2 3 2 1 1 1 2 1 1 1 2 3

pip(2)

2 1 1 1 2 1 1 1 2

pip(1)

1 1 1

pip(0)

base case → nothing

Ques:

PW SKILLS

Q5 : Print zig-zag (Already solved) pip

Input	Output
1	111
2	211121112
3	32111211123211121112 3
4	4321112111232111211123432111211123211121112 34

↓

$n \rightarrow T.C. = O(2^n)$

Traversing an array using recursion

PW SKILLS

Print all the elements of an array

```java
public static void print(int i, int[] arr){
    if(i==arr.length) return;
    System.out.print(arr[i]+" ");
    print(i+1,arr);
}
public static void main(String[] args) {
    int[] arr = {4,7,1,3,8,6,9};
    print(0,arr);
}
```

Skip a character

Remove all occurrences of 'a' from a string.

```
String s = "Raghav";
String ans = "Rghv";
```

T.C. = O(n)

skip(0, "Raghav", "")
↓
skip(1, "Raghav", "R")
↓
skip(2, "Raghav", "R")
↓
skip(3, "Raghav", Rg)
↓
skip(4, Raghav, Rgh)
↓
skip(5, Raghav, Rgh)
↓
skip(6, Raghav, Rghv)

PW SKILLS

Subsets

PW SKILLS

Print subsets of a string with unique characters.

Follow Up : **Do it for array as well**

{ 1, 2, 3 }

↓

{ {}, {1}, {2}, {3}, {1,2}, {1,3}, {2,3}, {1,2,3} }

power set is set of all subsets

[Leetcode 78]

Subsets

PW SKILLS

Print subsets of a string with unique characters.

Follow Up : **Do it for array as well**

String s = "abcd";

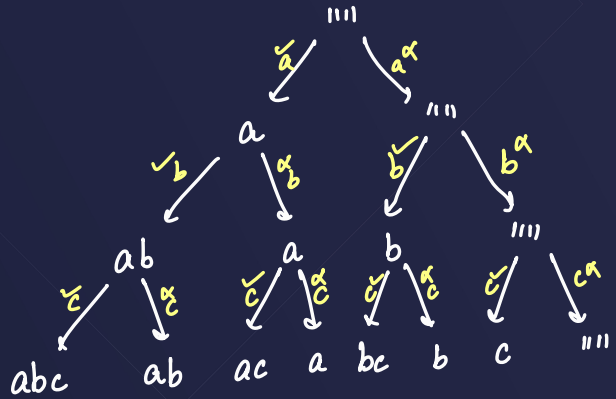
ϕ , a, b, c, d , ab , ac, ad , bc, bd, cd,

abc, abd , acd, bcd , abcd

[Leetcode 78]

$s = abc$

i



$2 + 4 + 8$

$tnc = 2^1 + 2^2 + 2^3 \ldots 2^n \simeq 2 \cdot 2^n \rightarrow$ **T.C. = $O(2^n)$**

Subsets of String (Code)

```java
static ArrayList<String> arr = new ArrayList<>(); // global
public static void printSubsets(int i, String s, String ans){
    if(i==s.length()){
        arr.add(ans);
        return;
    }
    printSubsets(i+1,s,ans); // not take
    ans += s.charAt(i);
    printSubsets(i+1,s,ans); // take
}
public static void main(String[] args) {
    String s = "abcd";
    arr = new ArrayList<>(); // reset
    printSubsets(0,s,"");
    System.out.println(arr);
}
```

Ques: (Homework)

Q6 : Print all increasing sequences of length k from first n natural numbers.

n = 6

k = 4

→

1 2 3 4

1 2 3 5

1 2 3 6

1 2 4 5

1 2 4 6

1 2 5 6

1 3 4 5

1 3 4 6

1 3 5 6

1 4 5 6

2 3 4 5

2 3 4 6

2 3 5 6

3 4 5 6

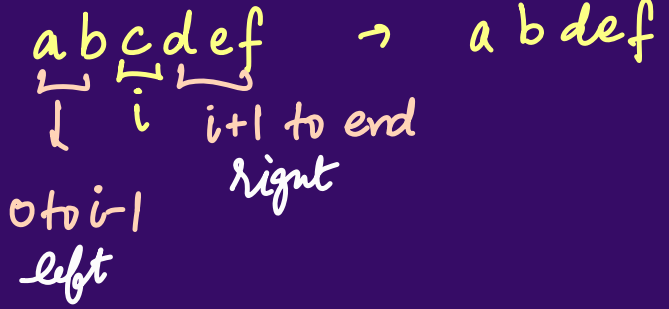
Permutations

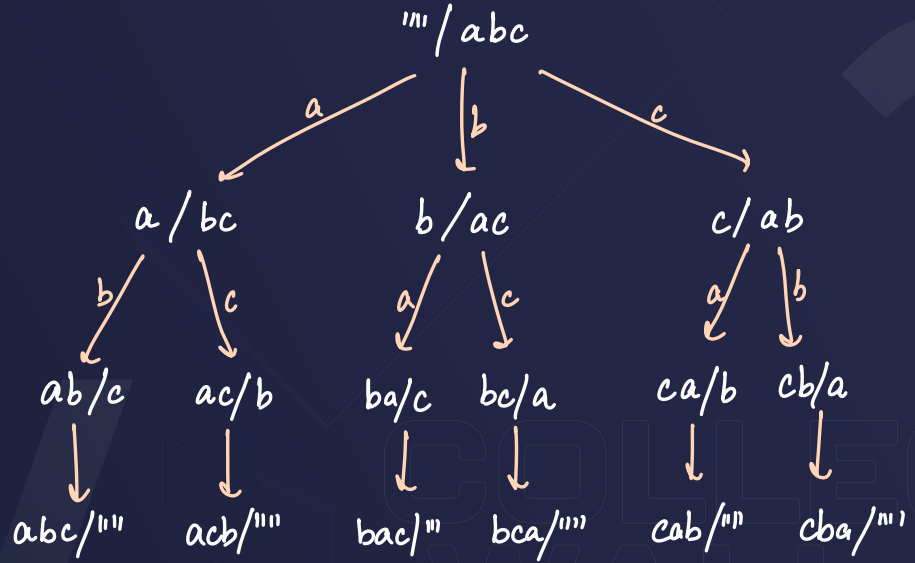
PW SKILLS

Finding all permutations of an string given all elements of the string are unique.

String s = abc , n! permutations

abc, acb, bac, bca, cab, cba




"" / abc
a
b
c
a / bc
b / ac
c / ab
b
c
a
c
a
b
ab / c
ac / b
ba / c
bc / a
ca / b
cb / a
abc / ""
acb / ""
bac / ""
bca / ""
cab / ""
cba / ""

```java
public static void printPermutations(String ans, String s){
    if(s.length()==0){
        System.out.println(ans);
        return;
    }
    for(int i=0;i<s.length();i++){
        char ch = s.charAt(i);
        String left = s.substring(0,i);
        String right = s.substring(i+1);
        printPermutations(ans+ch,left+right);
    }
}
public static void main(String[] args) {
    String s = "abcd";
    printPermutations("",s);
}
```

$T.C. = n(n-1)(n-2)\ldots$

$= n!$

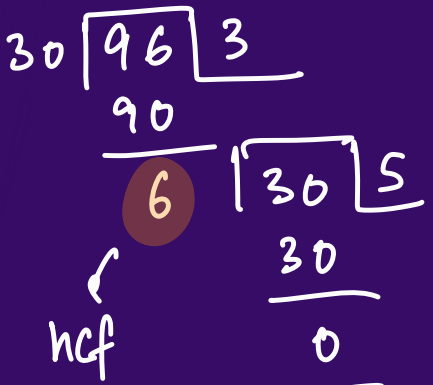
$T.C. = O(n!)$

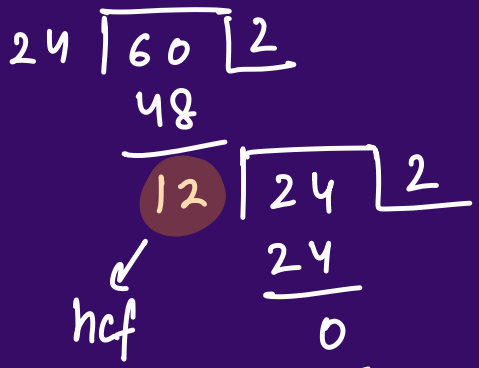
Greatest Common Divisor (HCF)

PW SKILLS

Calculate Greatest Common Divisor of two numbers.

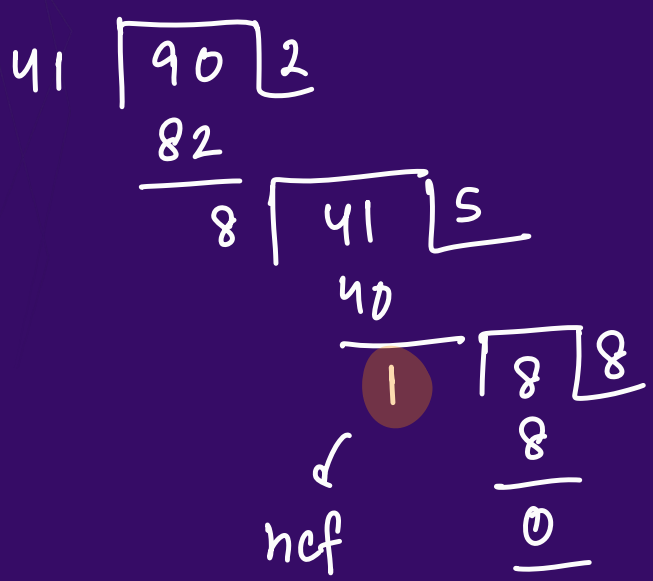
HCF(a,b) <= min(a,b)

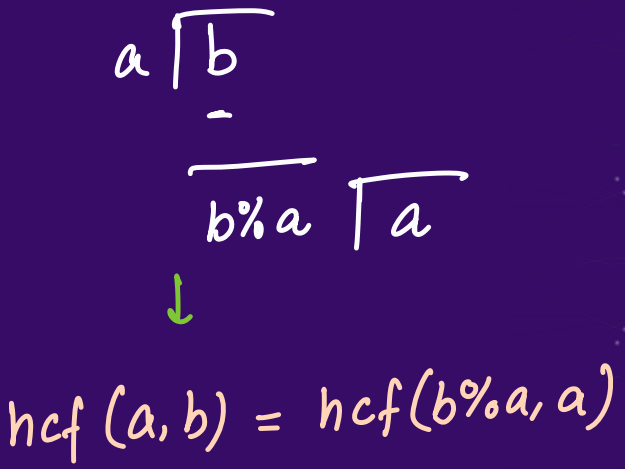




Greatest Common Divisor

Calculate Greatest Common Divisor of two numbers.





Greatest Common Divisor

PW SKILLS

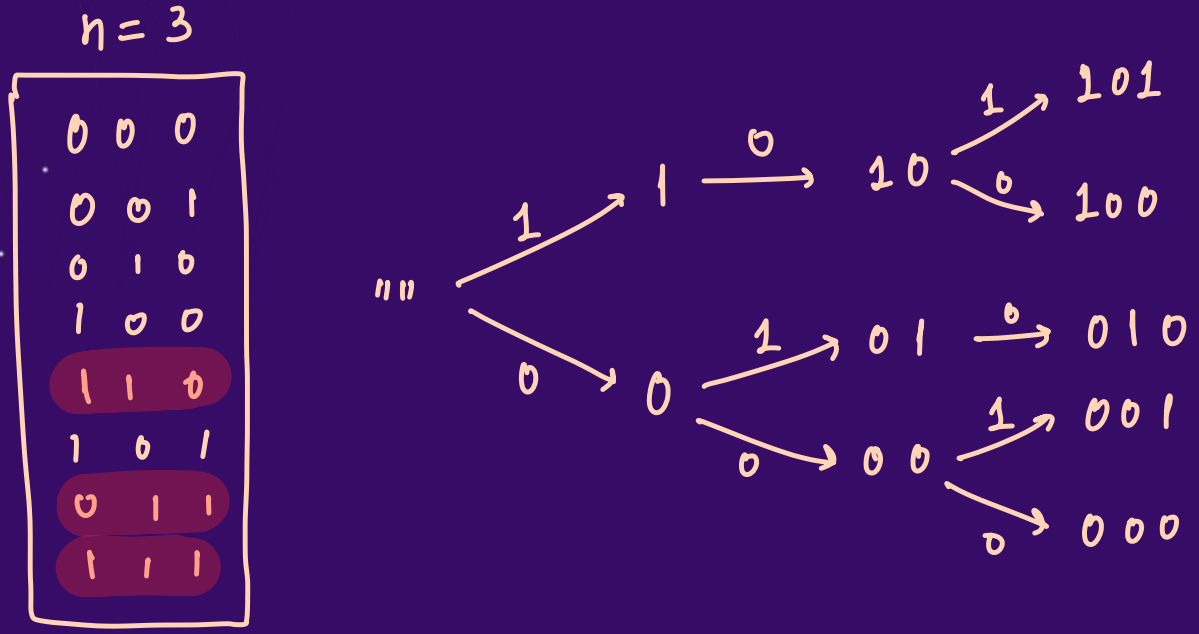
Calculate Greatest Common Divisor of two numbers.

gcd(96, 30)
↓
gcd(30, 96)
↓
gcd(6, 30)
↓
ans → 6

gcd(60, 24)
↓
gcd(24, 60)

Practice

Generate all binary strings of length n without consecutive 1's



Practice

PW SKILLS

Generate all binary strings of length n without consecutive 1's

```java
public static void printStrings(String s, int n){
    int m = s.length();
    if(m==n){
        System.out.println(s);
        return;
    }
    if(m==0 || s.charAt(m-1)=='0'){
        printStrings(s+1,n);
        printStrings(s+0,n);
    }
    else printStrings(s+0,n);
}
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int n = sc.nextInt();
    printStrings("",n);
}
```

Ques: [], {}, ()

pair of parantheses

Q2 : Generate Parentheses

n = 3 : We have 3 opening bkts & 3 closing brackets

→ ((()))

()()()

(())()

()(())

(()())

())(() → ✗

)()(() → ✗

[Leetcode 22]

Ques:

PW SKILLS

Q2 : Generate Parentheses

Wrong Approach → adding to left/right & enclosing

n = 1 ()

n = 2 ()() , (())

n = 3 ()()() , (())() , ()(()) , (()()) , ((()))

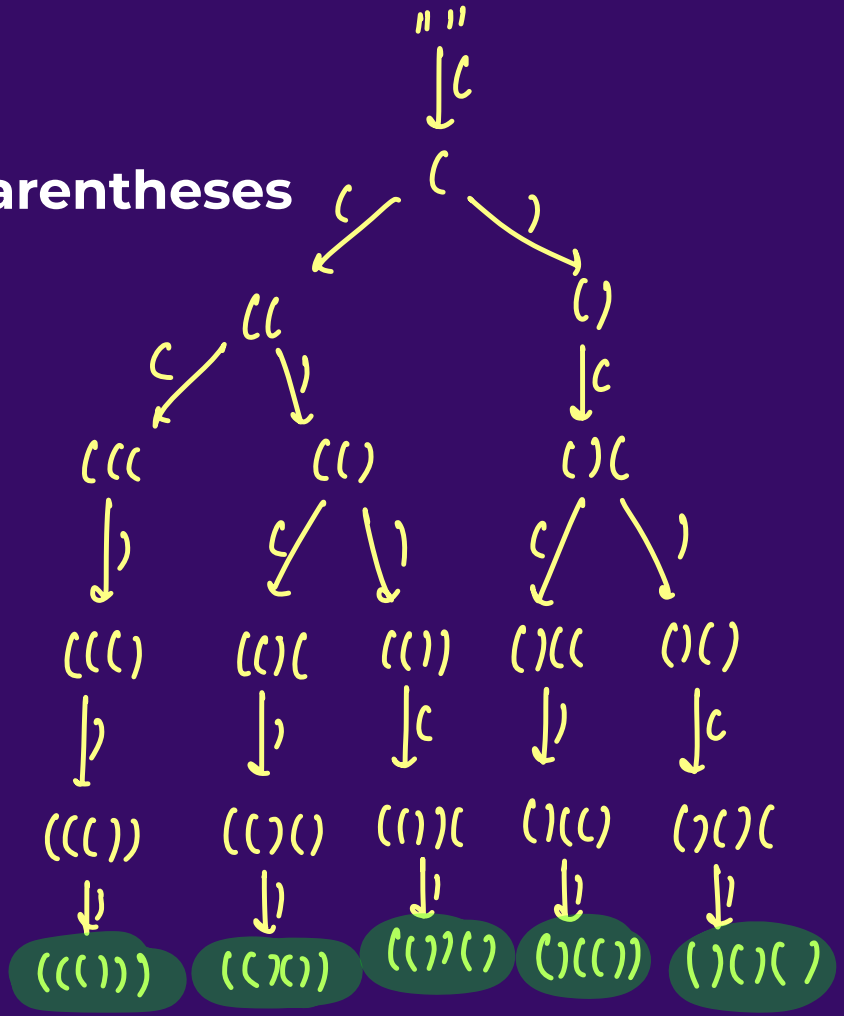
Hint : At any point, no. of closing bkts cannot be greater than no. of opening bkts

[Leetcode 22]

Ques:

Q2 : Generate Parentheses

n=3



[Leetcode 22]

Ques:

Q3 : Kth Symbol in Grammar

[Leetcode 779]

Ques:

PW SKILLS

Q4 : Count and Say

s = 3 3 2 2 2 5 1

↓ modify

ans = 2 3 3 2 1 5 1 1

s = 3 3 3 3 2 2 2 3 3

↓ modify

4 3 3 2 2 3

[Leetcode 38]

Ques:

PW SKILLS

Q4 : Count and Say

countandSay(n) is the way you "say" countandSay(n-1)

cas(1) = 1

cas(2) = 11

cas(3) = 21

cas(4) = 1211

cas(5) = 111221

[Leetcode 38]

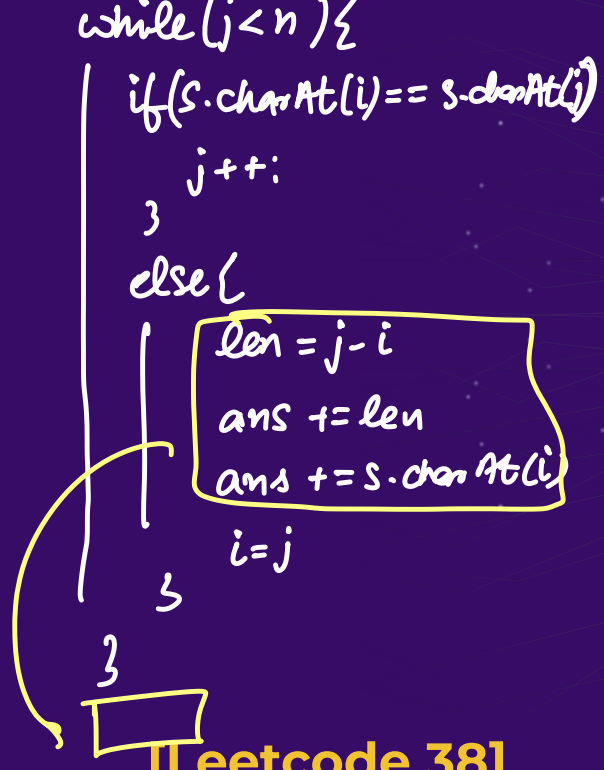
Ques:

PW SKILLS

Q4 : Count and Say

	0	1	2	3	4	5	6	7	8	9
s =	3	3	2	2	2	5	1	1	1	1
							i			j

ans = 2 3 3 2 1 5 4 1



[Leetcode 38]

THANK YOU