

LeetCode [Q.No.: 1 - 2064]

\Rightarrow Minimized maximum of Products distributed to any.

| M | T | W | T | F | S | S |
|-----------|---|---|---|---|---|-------|
| Page No.: | | | | | | YOUVA |
| Date: | | | | | | |

Ex ① :-

$\text{order} = \{ 14, 6 \}$ $n=6$.
 ↓ ↓ ↓
 warehouse 11 units of 6 units of Samsung.
 IPhone

Hi 6s 0 0 0 0 max = 11
Retail stores.

$$\frac{3i}{3i} \quad \frac{3i}{3i} \quad \frac{3i}{3i} \quad \frac{2i}{2i} \quad \frac{3s}{3s} \quad \frac{3s}{3s} \quad \text{max} = 3$$

$$\text{Ex } ② \quad \text{ans} = \{ 15, 10, 10 \} \quad n=7$$

$$o = (\pm)5$$

~~lo = 15~~
~~hi = 15 * 8 * 4~~ Stories = arr[i] / maxQ + 1 MaxAns = 8 * 6 * 5.
~~mid = 8 * 8 * 5~~ if (arr[i] % maxQ != 0)

⇒ Code :- Psum % (int [] arr , int n , int m , int max) {

int lo = 1, hi = mx;

```
int ans = 0;
```

```
while (lo <= hi) {
```

$$\text{int mid} = l_0 + (h_1 - l_0) / 2 ;$$

```
if (isPossible(mid, n, arr)) {
```

ans = mid ;

$$hi = mid - 1; j$$

new ♂ ←

1

sout (ans);

5. \Rightarrow Function :- Ps boolean isPossible(int max(), int n, int[] arr){
 int stores = 0;
 for (int i = 0; i < arr.length; i++) {
 if (arr[i] % max() == 0) stores += arr[i] / max();
 else stores += arr[i] / max() + 1;
 }
 if (stores > n) return false;
 else return true;
}

Leet Code [Q.No.: 153g]

→ K^{th} Missing Positive Number [Binary Search + Math] O(log n)

| M | T | W | T | F | S | S |
|-----------|---|---|---|---|-------|---|
| Page No.: | | | | | YOUVA | |

Ex (D):

arr = { 2, 3, 4, 7, 11 } → $K = 5$.

No. of missed elem till that index.

$$\text{missed} = \text{arr}[\text{mid}] - (\text{mid} + 1)$$

if (missed < k) lo = mid + 1;

else hi = mid - 1.

⇒ Observations: - The K^{th} missing No. is b/w $\text{arr}[\text{hi}]$ & $\text{arr}[\text{lo}]$.

$$K^{th} \text{ missing no.} = \text{arr}[\text{hi}] + \text{extra}$$

$$= \text{arr}[\text{hi}] + K - (\text{arr}[\text{hi}] - (\text{hi} + 1))$$

$$= \text{arr}[\text{hi}] + K - (\text{arr}[\text{hi}] - (\text{hi} + 1))$$

$$= K + (\text{hi} + 1).$$

or,

$K + \text{lo}$

$$K^{th} \text{ missing number} = \text{arr}[\text{hi}] + \text{extra}$$

$$\text{extra} = K - (\text{missed no.s till hi})$$

$$= K - (\text{arr}[\text{hi}] - (\text{hi} + 1))$$

No. is b/w $\text{arr}[\text{hi}]$ & $\text{arr}[\text{lo}]$.

⇒ Code :- Psvm {

int lo = 0, hi = arr.length - 1;

while (lo <= hi) {

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

int missed = arr[mid] - (mid + 1);

if (missed < k) lo = mid + 1;

else hi = mid - 1;

}

sout (K + lo);

}

5 / Sep. / 2024.

Recursion.

What & Why?

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

Recursion is not a D.S., it is an algo.

We believe in Recursion.

→ Recurrence Relations → Big Problem

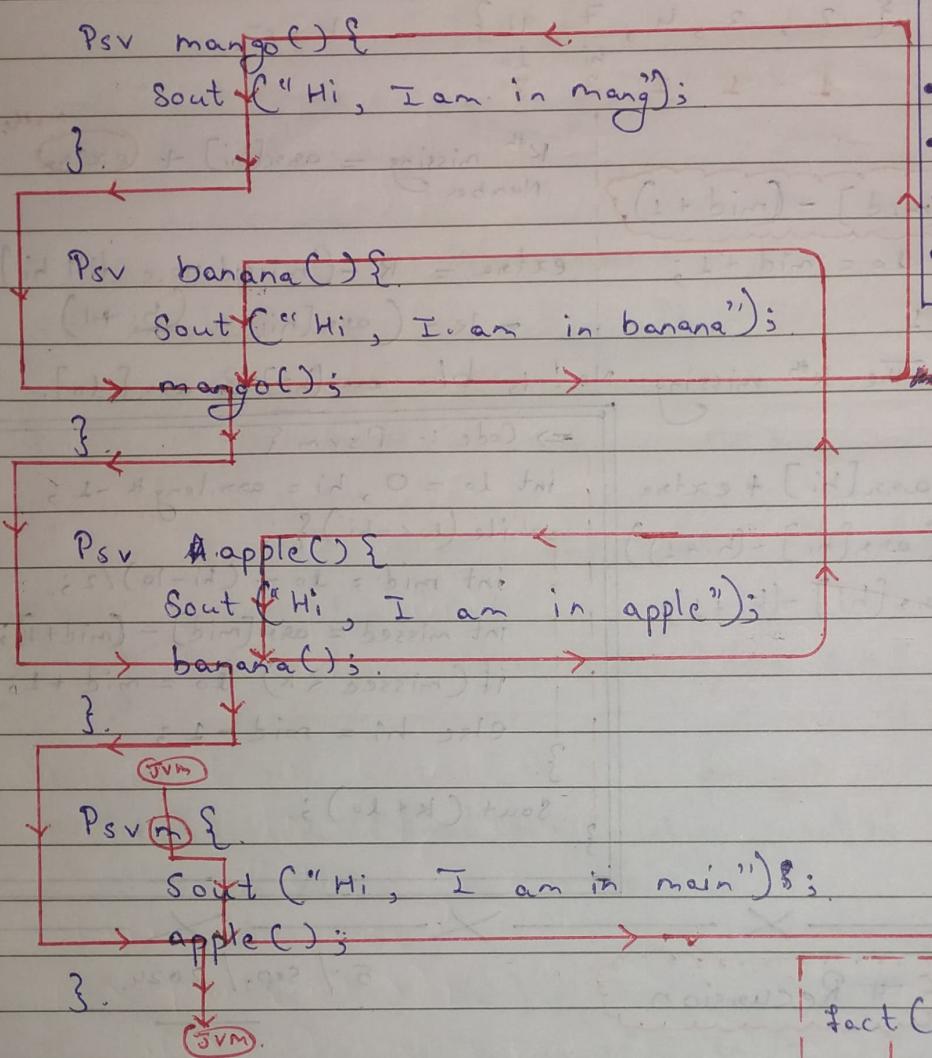
Using ↓
Smaller Subproblems.

(Method).

→ It's Nothing but Function calling it self.

⇒ Function Calls :-

⇒ Code :-



Output.

- Hi, I am in main.
- Hi, I am in apple.
- Hi, I am in banana
- Hi, I am in mango.

⇒ Rec Factorial Recurrence Relation:

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

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

$$6! = 6 \times 5!$$

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

$$8! = 8 \times 7!$$

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

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

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

$$f(x) = x!$$

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

$$\text{fact}(5) \leftarrow 120$$

$$5$$

$$5 \times \text{fact}(4) \leftarrow 120$$

$$4$$

$$4 \times \text{fact}(3) \leftarrow 120$$

$$3$$

$$3 \times \text{fact}(2) \leftarrow 120$$

$$2$$

$$2 \times \text{fact}(1) \leftarrow 120$$

$$1$$

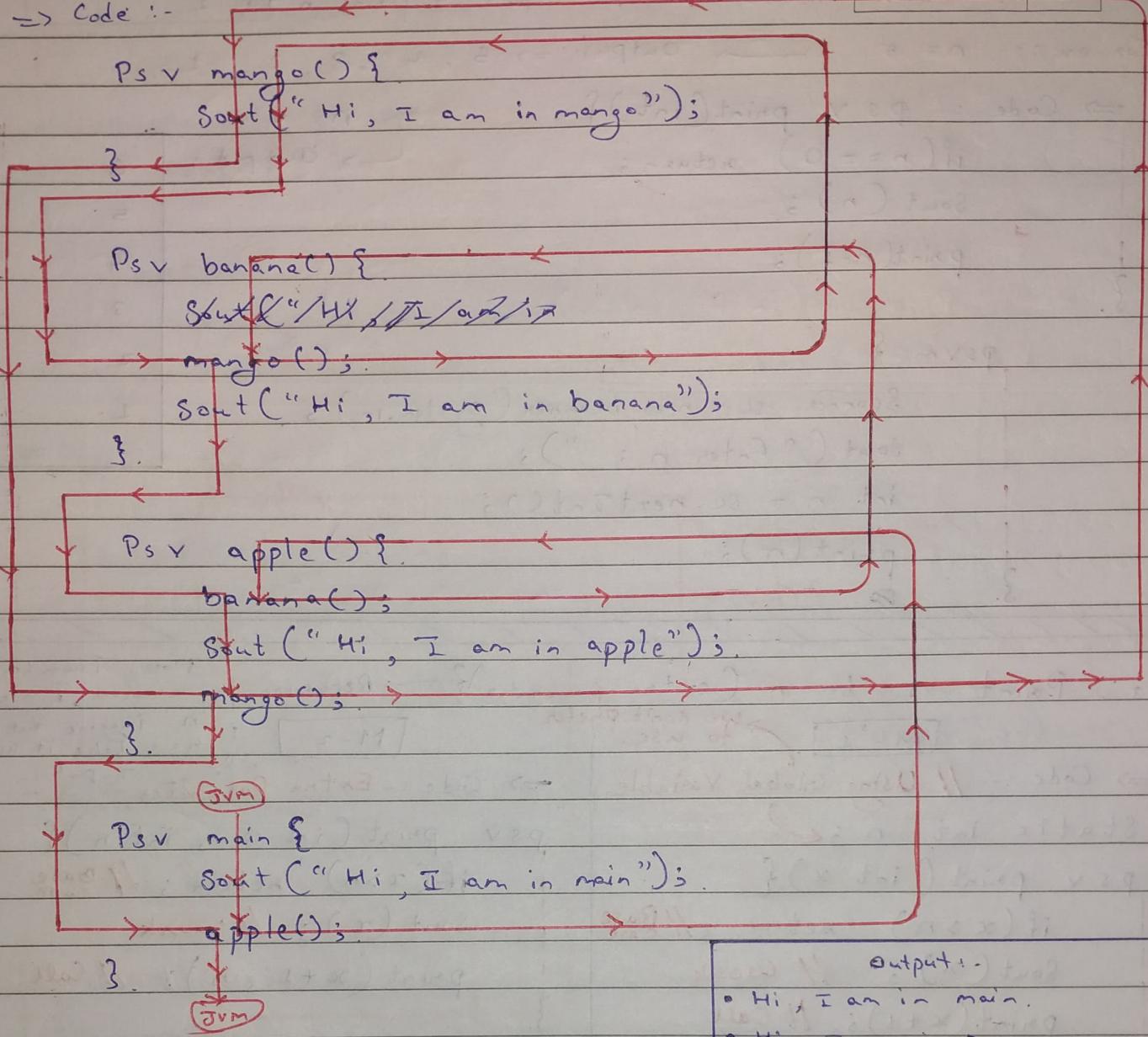
$$\text{base Case}$$

$$\# \text{Stoping point}$$

⇒ Function Calls :-

| M | T | W | T | F | S | S |
|-----------|---|---|---|---|---|-------|
| Page No.: | | | | | | YOUVA |
| Date: | | | | | | |

⇒ Code :-



Output:-

- Hi, I am in main.
- Hi, I am in mango.
- Hi, I am in banana.
- Hi, I am in apple.
- Hi, I am in mango.

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

$$\text{fact}(n) = n * \text{fact}(n-1)$$

⇒ Code :- Psv m { }

```
Scanner sc = new Scanner(System.in);  
Sout("Enter n: ");
```

```
int n = sc.nextInt();
```

```
Sout(fact(n));
```

```
}
```

→ Base Case :- Need to give stopping point else it will be in ∞ loop.

→ Call :- It is calling it self.

function:-

```
Psv int fact(int n) {
```

```
    if (n == 1) return 1; // Base Case
```

```
    int ans = n * fact(n - 1); // Call
```

```
    return ans;
```

```
}
```

8 / sep / 2004

M T W T F S S
Page No.:

YOUVA

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.

```
→ Code :- psv print(int){  
    if(n == 0) return;  
    sout(n);  
    print(n-1);  
}
```

n = 5

↳ Output :-

5

4

3

2

1

psvm {

Scanner sc = new Scanner(System.in);

sout("Enter n : ");

int n = sc.nextInt();

print(n);

}

ss.

Q3. Print 1 to n (extra parameter)

Note:- From here whenever I use 'n' imagine that I have take it as

M - 1

we don't prefer to use

M - 2

→ Code :- // Using Global Variable

Static int n;

```
psv print(int x){  
    if(x > n) return; // Base case  
    sout(x); // Work  
    print(x+1); // Call  
}
```

→ Code :- Extra Parameter

```
psv print(int x, int n){  
    if(x > n) return; // Base case  
    sout(x); // Work  
    print(x+1, n); // Call  
}
```

psvm {

n = sc.nextInt();

print(1);

}

psv n {

int n = sc.nextInt();

print(1, n);

}

⇒ Recursion :-

```
fun()  
{  
    base Case;  
    work;  
    Call;  
    work;  
}
```

```
fun()  
{  
    base Case;  
    Call;  
    work;  
    Call;  
}
```

Qn. Print 1 to n (After recursive call) M - 3.

→ Code :- Ps v print (int n) {

 if (n == 0) return; // Base Case.

 // sout (n); // work } → if you want to print 1st n to 1 & then 2 to n.

 print (n - 1); // Call

 sout (n); // Work,

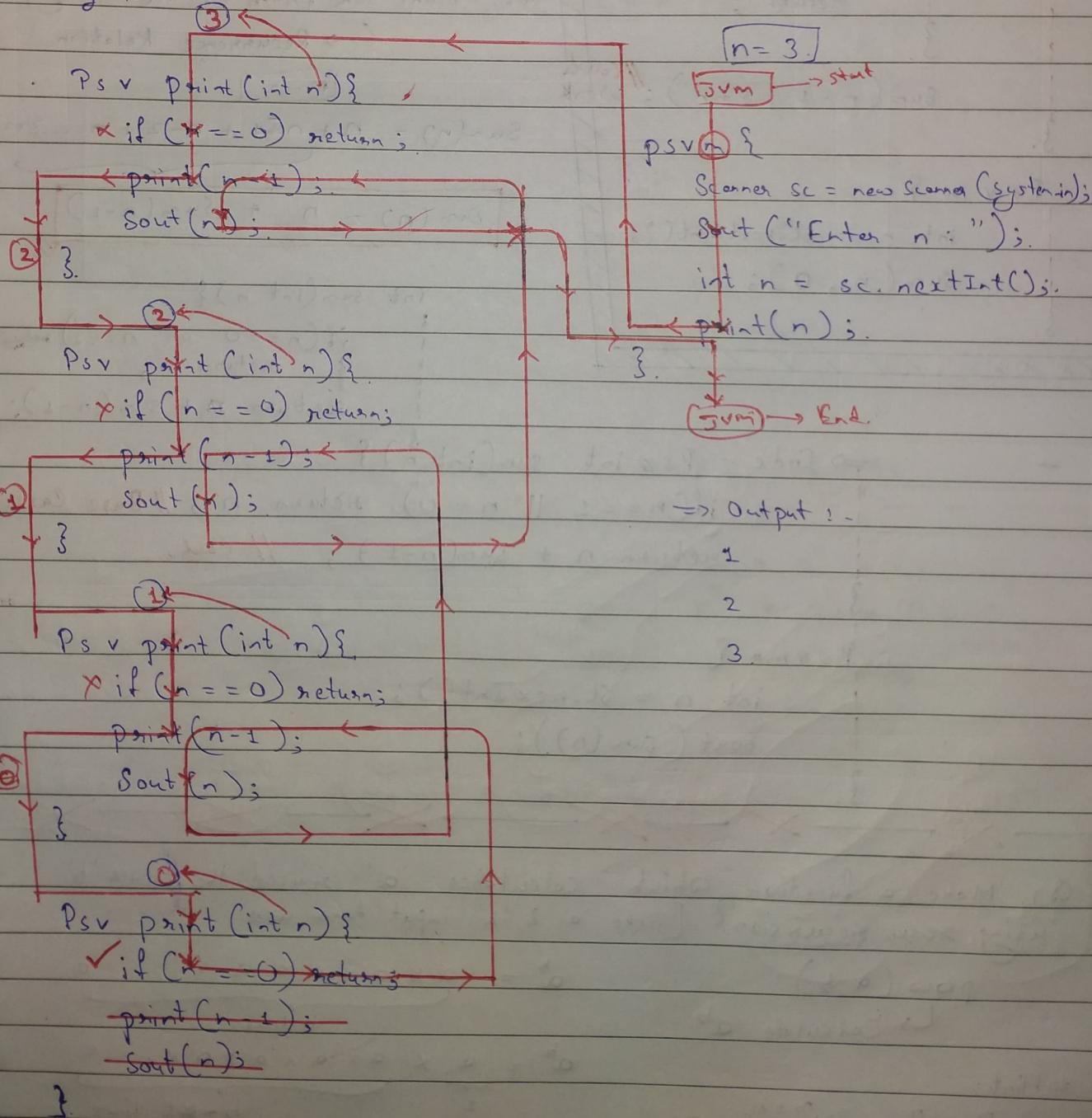
}

ps v main {

 int n = sc.nextInt();
 print (n);

}

Day Running



| M | T | W | T | F | S | S |
|-----------|-------|---|---|---|---|---|
| Page No.: | YOUVA | | | | | |
| Date: | | | | | | |

Q5. Print sum of from 1 to n [Parameterised] Recursion

| M | T | W | T | F | S | S |
|-----------|-------|---|---|---|---|---|
| Page No.: | YOUVA | | | | | |
| Date: | | | | | | |

$$n=5 \rightarrow \text{Sum} = 1 + 2 + 3 + 4 + 5. \quad \left. \begin{array}{l} \text{On} \\ 1+2+3+4+5 \end{array} \right\} 15 \quad \xrightarrow{\text{ans}}$$

using
for loop
{
 for (int i=1; i<=n; i++) {
 sum += i;
 }
}

$$\text{Sum} : (5, 0)$$

$$\downarrow \quad \text{Sum} : (4, 5)$$

$$\downarrow \quad \text{Sum} : (3, 9)$$

$$\downarrow \quad \text{Sum} : (2, 12)$$

$$\downarrow \quad \text{Sum} : (1, 14)$$

=> Code :- Ps v sum(int n, int s){

```

if (n==0) { //Base Case
    Sout(s);
    return;
}

```

sum(n-1, s+n); // Call & Work.

}

Ps v m {

int n = sc.nextInt();

sum(n, 0);

}

Q6. Print sum from 1 to n (ReturnType).

↳ Recurrence Relation.

[M-L]

$$\text{Sum}(n) = n + \underbrace{n-1 + n-2 + \dots + 2}_{\text{Sum}(n-1)}$$

$$\boxed{\text{Sum}(n) = n + \text{Sum}(n-1)}$$

int sum(int n){

if (n==0 || n==1) return n;

return n + sum(n-1);

=> Code :- Ps v int sum(int n){

if (n==1 || n==0) return n; // Base Case.

return n + sum(n-1); // call,

}

Ps v m {

int n = sc.nextInt();

Sout(sum(n));

}

Q7. Make a function which calculates 'a' raised the power 'b' using ~~loop~~ recursion! [Take 'a' & 'b' input from user.]

pow(a,b)

↳ Calculate a^b

$$a^b = \underbrace{axaxaxa \dots a}_{b \text{ times}}$$

$$a^b = \underbrace{a \times axaxaxa \times \dots a}_{2 \text{ times}} \quad \underbrace{a \times a \times a \times \dots a}_{b-2 \text{ times}}$$

Hint:-

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

$$\{ \text{pow}(a,b) = a * \text{pow}(a,b-1) \}$$

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

| M | T | W | F | S |
|-----------|---|---|---|-------|
| Page No.: | 5 | 6 | 7 | YOUVA |

→ Code :- Ps8. int pow(int a, int b){
 if (b == 0) return 1;
 return a * pow(a, b-1);}

psvm {

if (a > sc.nextInt() || Enter base!);

if (b > sc.nextInt());

int a = sc.nextInt(); // [Enter base]

int b = sc.nextInt(); // [Enter power].

cout (a + " raised to the power " + b + " is " + pow(a, b));

3.

Q.8. Power function [logarithmic].

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

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

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

$$2^1 = 2 * 2^0$$

Very Slow
64 calls [Beker & slow].# Hint :- $x^{m+n} = x^m * x^n$.
 $a^b \rightarrow T.C = O(\log_2 b)$.

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

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

$$2^{16} = 2^8 * 2^8. \quad a^b = (a^{b/2})^2$$

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

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

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

$$2^1 = 2^0 * 2^0$$

$$6 \text{ calls.} = \log_2 64$$

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

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

pow(a, b) = ans * ans;

↓

Wrong:
[M-2]

$$2^5 = 2^2 * 2^2 * 2$$

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

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

Extra

when 'B' is odd.

→ Code :- ps. int pow2(int a, int b){

if (b == 0) return 1;

int ans = pow2(a, b/2);

if (b % 2 == 0) return ans * ans;

else return ans + ans * a;

3.

psvm { int a = sc.nextInt();

int b = sc.nextInt();

cout ("a + " raise to the power " + b + " is " + pow2(a, b));

$$(3^4)^2 * 3$$

$$(3^2)^2 * 3$$

$$(3^1)^2 * 3$$

Base Case. $\leftarrow (3^0)^2 * 3$.

Q9. Write a function to calculate the n^{th} Fibonacci Number using recursion.

| M | T | W | T | F | S | S |
|-----------|---|---|---|---|---|-------|
| Page No.: | | | | | | |
| Date: | | | | | | YOUVA |

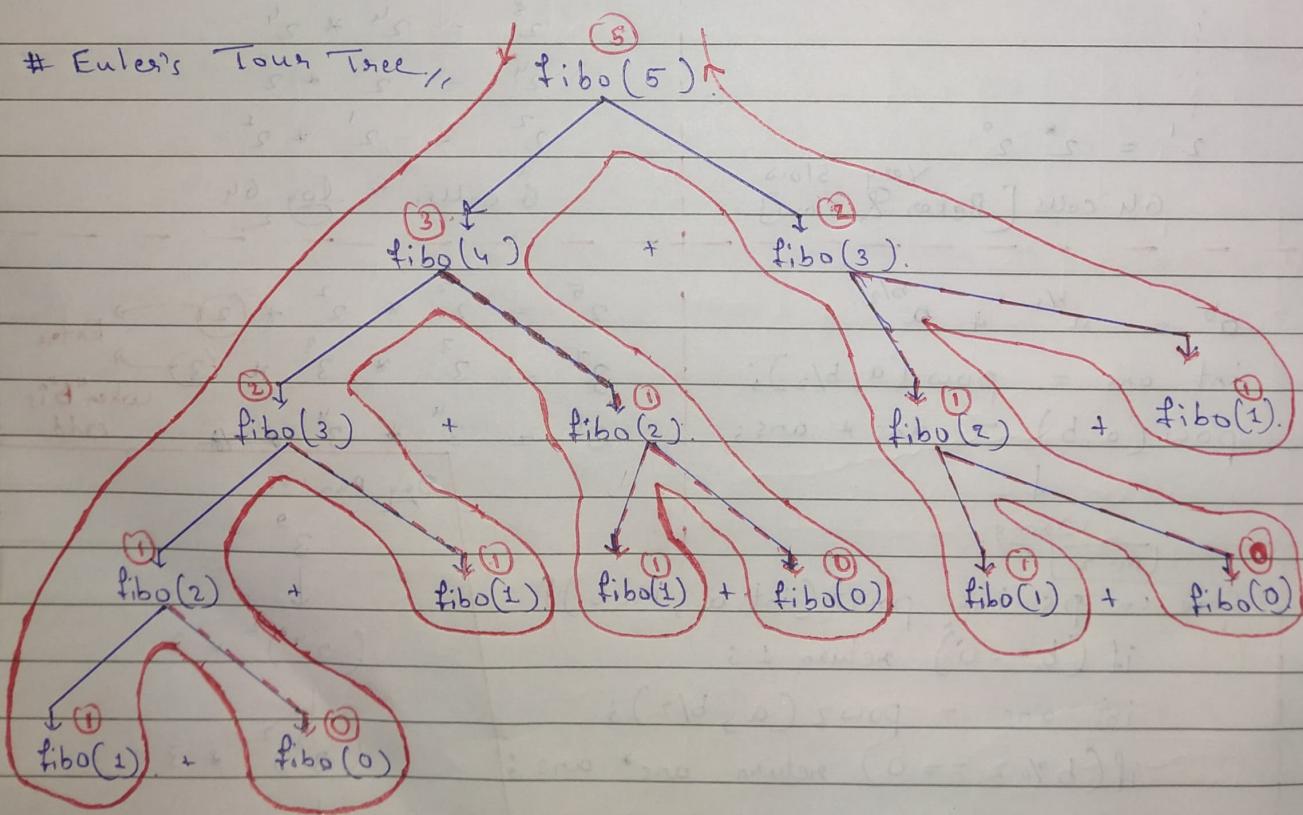
$n = 0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10 \ 11$
 0 1 1 2 3 5 8 13 21 34 55 89 ...

$$\{ \text{fibo}(n) = \text{fibo}(n-1) + \text{fibo}(n-2) \}$$

Base Case } \rightarrow if ($n \leq 1$) return n ;

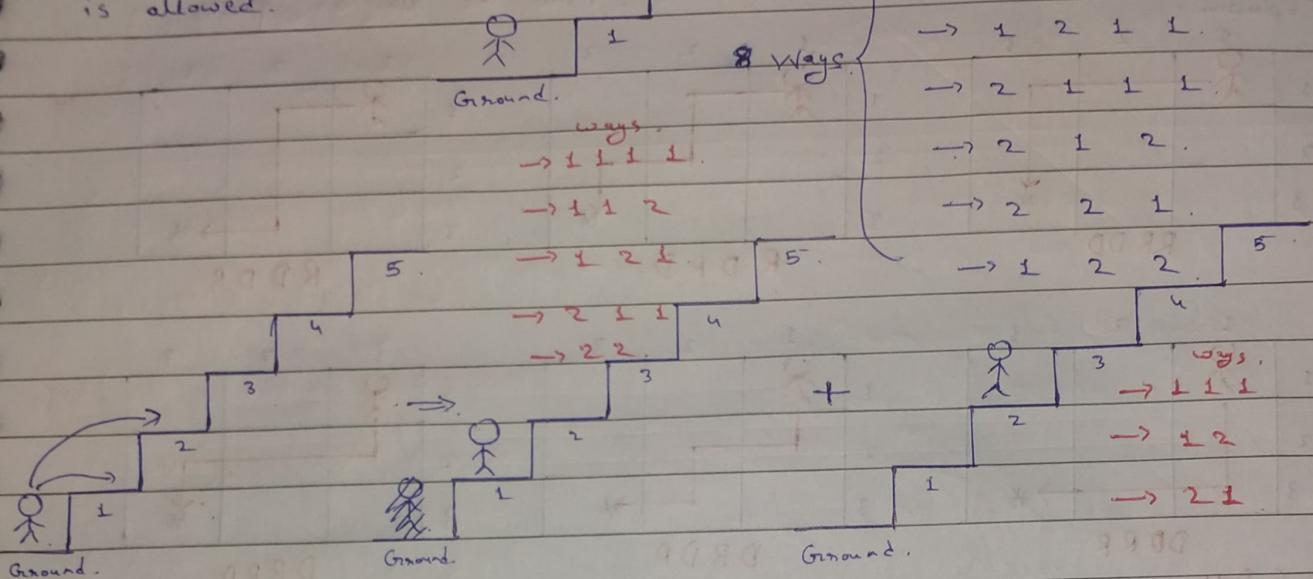
```
→ Code :- ps int fibo( int n ) {
    if( n <= 1 ) return n;
    return fibo(n-1) + fibo(n-2);
}

ps * m {
    int n = sc.nextInt();
    sout( fibo(n) );
}
```



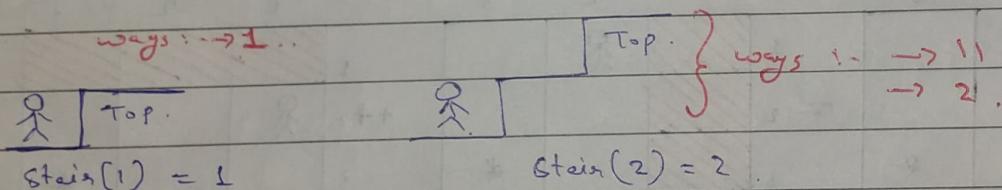
Stair Path.

$n = 5 \rightarrow \text{ways} = 8$
 ↳ Find no. of ways to reach n^{th} stair if 1 or 2 jump at a time is allowed.



$$\begin{aligned} \text{Stair}(5) &= \text{stair}(4) + \text{stair}(3). \\ \text{stair}(n) &= \text{stair}(n-1) + \text{stair}(n-2). \end{aligned}$$

Exactly same like a fibonacci



$\Rightarrow \text{Code: } \text{int stair(int } n\text{)}\}$

```

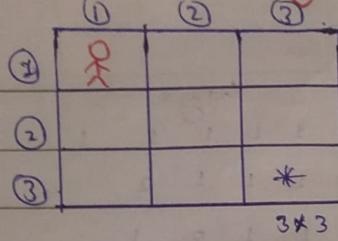
if ( $n <= 2$ ) return  $n$ ;
// if ( $n == 1$ ) return 1;
// if ( $n == 2$ ) return 2;
return stair( $n - 1$ ) + stair( $n - 2$ );
}
  
```

You can use this two also instead of first both are same only.

```

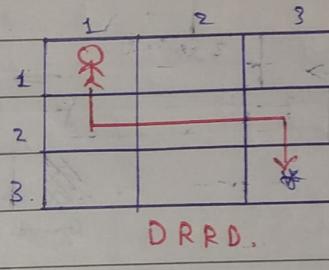
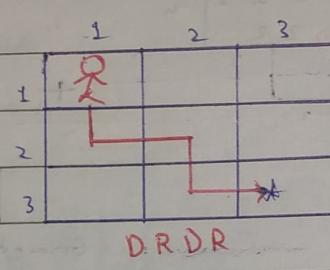
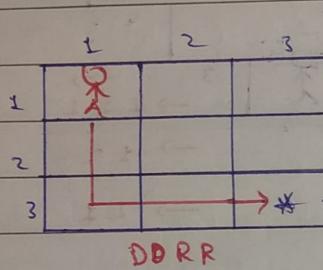
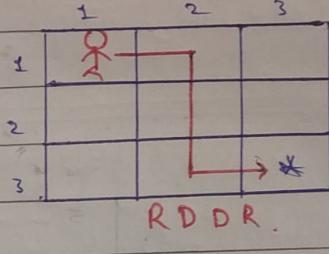
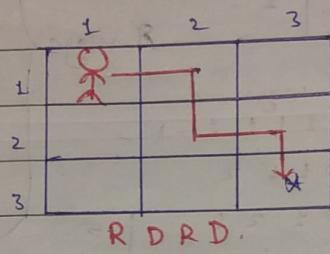
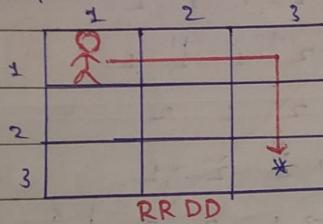
psvm {
    int n = sc.nextInt();
    sout(stair(n));
}
  
```

Maze Path [Very Imp].



Given a $m \times n$ 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.

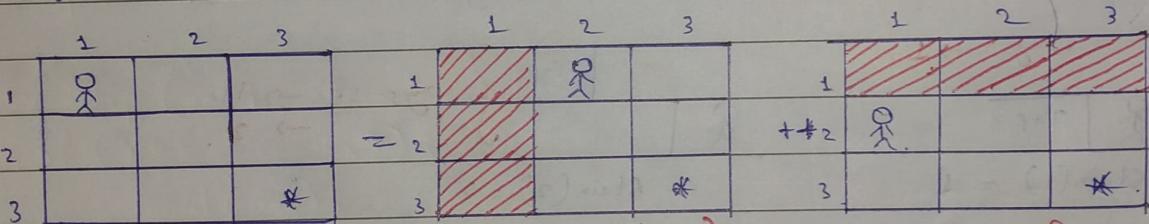
Explanation :-



$\Rightarrow 6$ Paths,

\Rightarrow Logic :-

$M - 1$

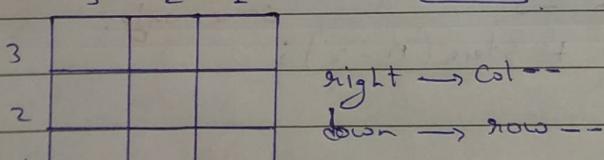


$$(1,1) \rightarrow (3,3) = (1,2) \rightarrow (3,3) + (2,1) \rightarrow (3,3),$$

$$\text{total ways} = \text{rightways.} + \text{downways.}$$

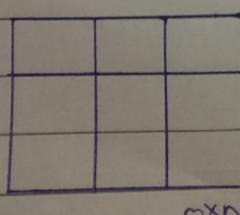
Base Case \rightarrow if ($\text{row} = m$ or $\text{col} = n$) return 1;

$M - 2$

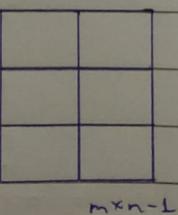


$$(m, n) \rightarrow (1, 1)$$

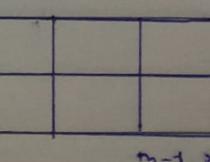
$$(1, 1) \xrightarrow{\text{on}} (m, n)$$



=



+



$m-1 \times n$

[M - 1]

```

⇒ Code :- Ps int maze(int row, int col, int m, int n) {
    if (col == n || row == n) return 1;
    // if (row == m && col == n) return 1;
    // if (row > m || col > n) return 0;
    int rightWays = maze(row, col + 1, m, n);
    int downWays = maze(row + 1, col, m, n);
    return rightWays + downWays;
}

```

| Page No.: | YOUVA |
|-----------|-------|
| Date: | |

PsVm {

```

int n = sc.nextInt();
int m = sc.nextInt();
//Sout(maze(1, 1, m, n));
Sout(maze2(m, n));
}

```

[M - 2]

```

ps int maze2 (int row, int col) {
    if (col == 1 || row == 1) return 1;
    int rightWays = maze2 (row, col - 1);
    int downWays = maze2 (row - 1, col);
    return rightWays + downWays;
}

```

Pre In Post :-

```

fun() {
    base case
    work
    (func)
    work
    (func)
    work.
}

```

⇒ Code :- PsV pip(int n) {

```

if (n == 0) return;
Sout(n + " ");
pip(n - 1);
Sout(n + " ");
pip(n - 1);
Sout(n + " ");
}

```

PsVm {
 pip(3);
}

⇒ You can do this using Call stack technique. we will use different method.

Day Run :-

1 Pip(3)



3 pip(2) 3 pip(2) 3.

1 pip(2)

2 pip(1) 2 pip(1) 2.

→ pip(1)

1 pip(0) 1 pip(0) 1.

pip(0)

↳ Base Case → Nothing

Output ⇒ 3 2 1 1 1 2 1 1 1 2 3 2 1 1 1 2 1 1 1 2 3.

Traversing an array using recursion.
→ Print all the elements of an array.

| M | T | W | T | F | S | S |
|-----------|-------|---|---|---|---|---|
| Page No.: | | | | | | |
| Date: | YOUVA | | | | | |

⇒ Code :- psv print(int i, int [] arr){

if (i == arr.length) return;

sout (arr[i] + " ");

print (i + 1, arr);

}

psvm {

int [] arr = {4, 7, 1, 3, 8};

print (0, arr);

}

Skip a Character

↳ Remove all occurrences of 'a' from a string.

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");

⇒ Code :- psv print(int i, String s){

if (i == s.length()) return;

sout (s.charAt(i));

print (i + 1, s);

}

psv skip (int i, String s, String ans){

if (i == s.length()) {

sout (ans);

return;

if (s.charAt(i) != 'a') ans += s.charAt(i);

skip (i + 1, s, ans);

}

psvm {

String s = "Raghav";

if print (0, s);

skip (0, s, "");