Q⇒ Given a value n (positive integer), print the following pattern on the screen.

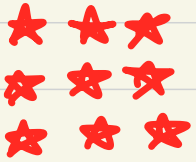
Ex⇒ n = 4

ans→

task is to print n stars in one line.

Ex→ n = 3

ans→



Ex→ n = 6

We have to print a (n x n) grid with all *

observation - 1

for a given value of n, we print n rows.

observation 2

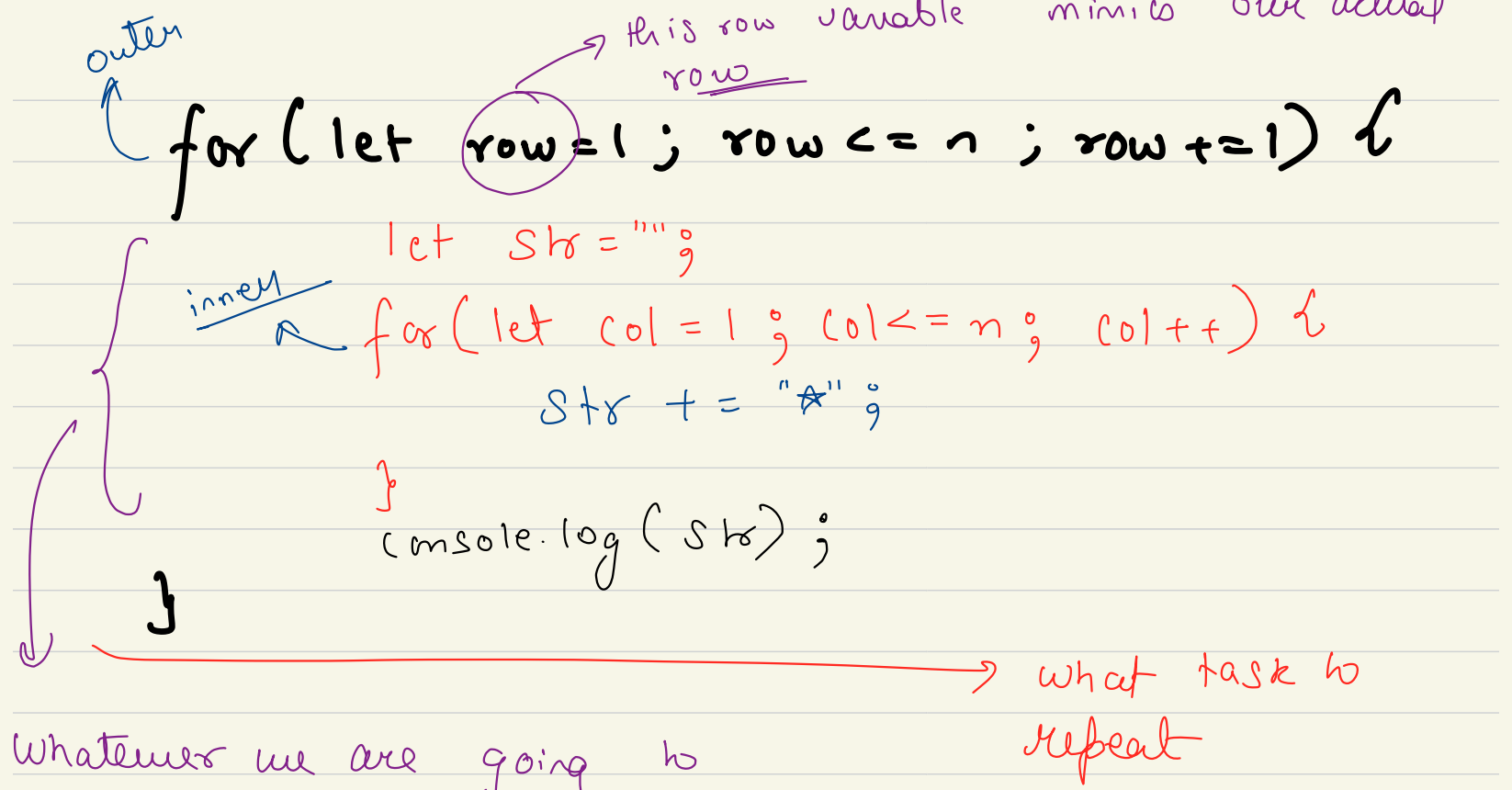
for a given row, we print n * in that row.

We have to do some task for each row.

and the task is the print n stars in that row.

How many rows? → (n)

Here the element of repetition is task, we repeatedly do the same task on each row.


outer
this row variable mimics our actual row
for (let row=1 ; row <= n ; row+=1) {
let str = "" ;
inner
for (let col = 1 ; col <= n ; col++) {
str += "*" ;
}
console.log (str) ;
}
what task to repeat
Whatever we are going to write inside the loop, JS will be repeating that again & again.

How can we build a string having "★★....★" n stars.

n = 4 → ★★★★

let stars = ""; → empty string.

How to repeat this process n times ?,?

↓

using loops

stars += "★" → stars = stars + "★"

"" + "★" → "★"

stars += "★" → stars = stars + "★"

↳ "★" + "★" → "★★"

stars += "★" → "★★★"

⋮

stars += "★" → "★★★★.....★" (n times)

→ (+) → addition operation

2 + 3 → 5

left + Right →
- "Sanket" + 10 → "Sanket" + "10" = "Sanket10" (Conversion)
- "abc" + "def" → "abcdef"

if any one either left or right or both are Strings then, it converts the other operand as a string also & then joins them.

This joing process of 2 Strings is called Concatenation.

In string concatenation, if we have both operands as strings then we just join them together & get a new string.

Eg → "abc" + "def" → "abcdef"

Eg → 2 + 3 → 5

"2" + "3" → "23"

a += b
↳ a = (a + b)

+= → this also does string concatenation

Note

```
let x = "abc"
let y = "def"
let z = "xyz"
```

let z = x + y

"abcdef" → new string

z += x →

"xyzabc"

String concatenation always create new strings & doesn't update old strings.

↳ Reason → In JS strings are immutable (cannot modify themselves)

```
function pattern(n) {
    // logic for printing the pattern
    // write a loop to repeat a task for n rows
    for(let row = 1; row ≤ n; row += 1) {
        // task
        // the task is to concatenate "*" n times
        let str = "";
        for(let col = 1; col ≤ n; col += 1) {
            str += "*";
        }
        console.log(str);
    }
}
```

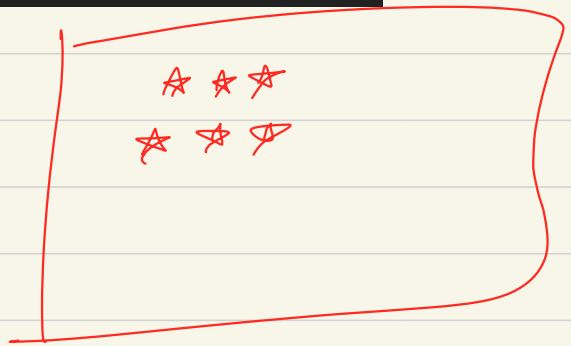
n = 3

row = 1 2 3 col = 1 2 3 4

str = "★★★"

Nested loops

↳ loop inside a loop



Q⇒ Given a value n (positive integer), print the following pattern on the screen.

Ex⇒ n = 4

ans→

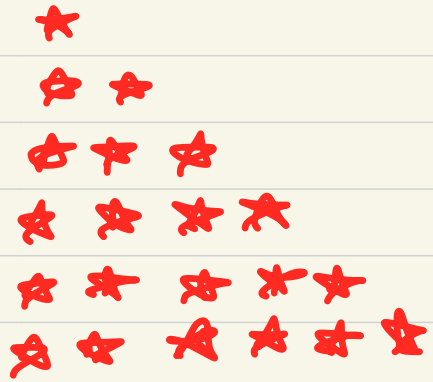
1) ★ ← current row = 1
2) ★ ★
3) ★ ★ ★
4) ★ ★ ★ ★

Ex→ n = 3

ans→

★
★ ★
★ ★ ★

Ex→ n = 6



this problem is a lot similar to the last problem in the sense that, here also we have to repeat a task (print some stars) for each row.

row = 1 2 3 4

```
for(let row=1; row<=n; row+=1){
    let str="";
    for(let col=1; col<=row; col+=1){
        str += "*";
    }
    console.log(str);
}
```

whatever logic we write will be repeated n times

Let's say

we are at row no. 4,

how many stars do we need ?? → 4

```
let str = "";
```

```
str += "*"
str += "*"
⋮
```

how many times of concatenation ?!

Should we repeat the operation 4 times

general sol^n → for any given row no, we have to repeat row no. times

```
for(let row=1; row<=n; row+=1) {
    let str="";
    for(let col=1; col<=row; col+=1) {
        str += "*";
    }
    console.log(str);
}
```

```
function pattern(n) {
    // logic for printing the pattern
    // write a loop to repeat a task for n rows
    for(let row = 1; row ≤ n; row += 1) {
        // task
        // the task is to concatenate "*" n times
        let str = "";
        for(let col = 1; col ≤ n; col += 1) {
            str += "*";
        }
        console.log(str);
    }
}
```

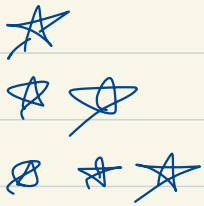
```
function pattern(n) {
    // logic for printing the pattern
    // write a loop to repeat a task for n rows
    for(let row = 1; row ≤ n; row += 1) {
        // task
        // the task is to concatenate "*" n times
        let str = "";
        for(let col = 1; col ≤ row; col += 1) {
            str += "*";
        }
        console.log(str);
    }
}

pattern(9);
```

n = 3

row = 1 2 3 4 5 6 7 8 9

col = 1 2 3 4 5 6 7 8 9

str = "* * *"



Q⇒ Given a value n (positive integer), print the following pattern on the screen.

Ex⇒ n = 4

ans→

1) _ _ _ ★

2) _ _ ★ ★

3) _ ★ ★ ★

4) ★ ★ ★ ★

Ex→ n = 3

ans→

_ _ ★

_ ★ ★

★ ★ ★

→ this is a Space

Ex→ n = 6

_ _ _ _ _ ★

_ _ _ _ ★ ★

_ _ ★ ★ ★

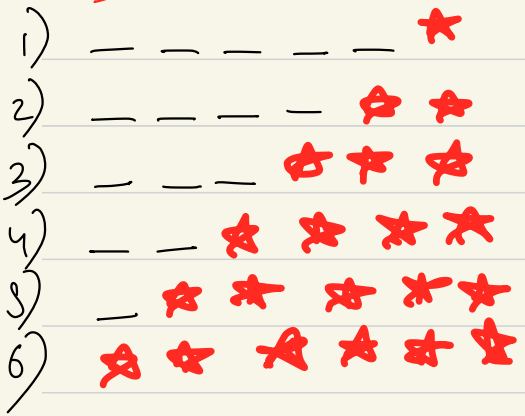
_ ★ ★ ★ ★

_ ★ ★ ★ ★ ★

★ ★ ★ ★ ★ ★

→ for every row we have to print some spaces followed by some *.

n = 6



Row	Spaces	Stars
1	5	1
2	4	2
3	3	3
4	2	4
5	1	5
6	0	6

Then, for any row no. → i → we need

stars → i spaces → n - i

outer

```
for (let row = 1 ; row <= n ; row += 1) {
    let str = "" ;
    let spaces = n - row ;
    for (let j = 1 ; j <= spaces ; j += 1) {
        str += " " ;
    }
    let stars = row ;
    for (let col = 1 ; col <= stars ; col += 1) {
        str += "*"
    }
    console.log(str)
}
```

no. of spaces in the curnt row

inner1

inner2

When this for ends after this we need one more loop for stars

This logic will be repeated

for any row no. i

→ we first need to print (n-i) spaces

→ then we need to print i stars.

row = 3 (1)

n = 7 (2)

_ _ _ _ ☆☆☆

Spaces → 7 - 3 = 4 spaces

Stars → 3 stars

let str = "" → this str variable will be having all the content of current row.

str += " "
str += " "
⋮
str += " "
} (n - row)
↳ we need a loop that goes for (n - row) no. of lines

str = "_ _ _ _"
str += "☆"
str += "☆"
⋮
} (row) → we need one more loop that goes for (row) no. of lines

after you've concatenated Spaces then and after then only you will add "*".

```
function pattern(n) {
    // we will write a loop which will repeat some ta
    for(let row = 1; row ≤ n; row += 1) {
        // inside this loop of row, we will do the ta
        // Task - <some spaces><some stars>
        // n = 7, row = 3, "    **" → this string w
        // So lets build the spaces first
        let str = "";
        let spaces =  n - row;
        // loop using which we will concatenate (n-ro
        for(let j = 1; j ≤ spaces; j += 1) {
            str += " ";
        }
        // after the above loop ends, we will be havi
        // ex: n = 7, row = 3, str = "    "
        // loop to concatenate stars
        let stars = row;
        for(let col = 1; col ≤ stars; col += 1) {
            str += "*";
        }
        // after the end of the above loop we have bo
        // stars in the string
        console.log(str); // print the string
    }
}
```

n = 3

row = 1 2 3

str = " — * * "

j = 1 2

Spaces = 3-2 = 1

Stars = 2

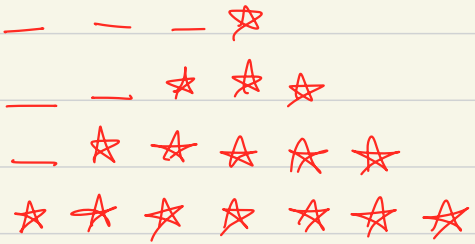
col = 1 2 3

— — *

— * *

$n = 4$

Q 1



Q 2

$n = 5$