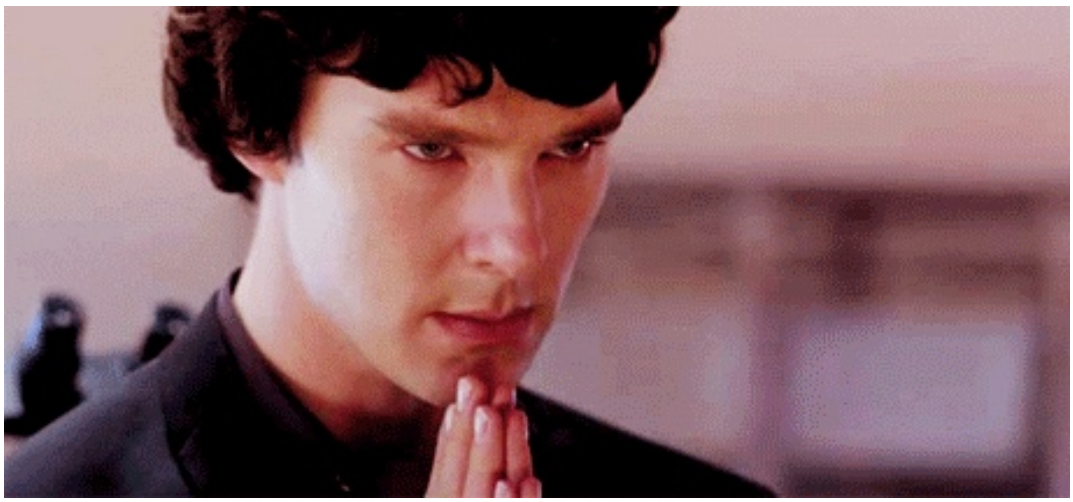
IMP

enjoy the song!!

Any Sherlock fans?



Agenda

① Solve 11 problems

Q1) Given N as input, print N stars?

Input : 4

* * * *

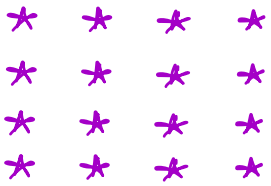
Input : 3

* * *

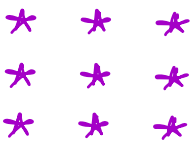
```
int n= scn.nextInt();
for (int i=1; i<=n; i++) {
    sop("*");
}
```

2) Given N as input, print N * N square

Input: 4



Input: 3



```
int n = scn.nextInt();
for (int row = 1; row <= n; row++) {
        for (int col = 1; col <= n; col++) {
                sop("*");
        }
        sopln();
}
```

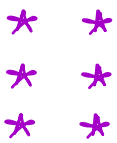
[n = 3]

Row	Col	Output
1	1, 2, 3	*** ↵
2	1, 2, 3	*** ↵
3	1, 2, 3	*** ↵

3) Given Input N & M, print a rectangle N*M

Input = 5, 4

Input = 3, 2



```
int n = scn.nextInt();
for (int row = 1; row <= n; row++) {
        for (int col = 1; col <= m; col++) {
                Sop ("*");
        }
        Soplnl();
}
```

Staircase pattern

Input 4

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

Input 3

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

Sherlocks tabular method

↳ Represent the output as a table

Row	Stars
1	1
2	2
3	3
:	:
n	n

Row == stars

↑ outer loop.

```
for (int row = 1 ; row <= n ; row++) {
    for (int st = 1 ; st <= row ; st++) {
        sop("*");
    }
    sopln();
```

}

5) Given N as input, print the following pattern

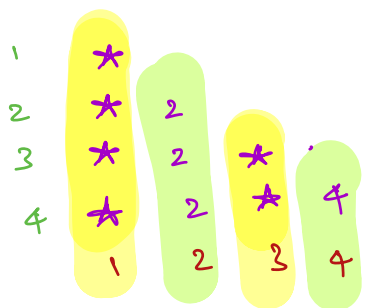
Input = 4

```
*
* 2
* 2 *
* 2 * 4
```

Input = 3

```
*
* 2
* 2 *
```

Observation



→ odd value of col. we print *

→ even value of column. we print col.

```
for (int row=1; row <= n; row++) {
    for (int col=1; col <= row; col++) {
        if (col % 2 == 0) {
            // even
            sop (col);
        }
        else {
            // odd
            sop ("*");
        }
```

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

// patterns with spaces

6) Given N as input, print the following pattern

Input : 4

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

Input : 3

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

Observe

n = 4	x = 2
n = 5	x = 3
n = 3	x = 1

number of spaces

n - 2

```
for ( int row=1;  row <=n ;  row ++) {
        Sop (" *");
        for (Pnt sp = 1 ;   sp <= (n-2) ;  sp++) {
                Sop ("  ");
        }
        Sop (" *");
        Soplin ();
}
```

//9:49 → 9:56

---

7) Given N as input, print the following pattern

n = 4

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

n = 3

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

Sherlock's tabular Method n=4

row		stars		sum
1	+	4	=	5 (n+1)
2	+	3	=	5
3	+	2	=	5
4	+	1	=	5

row + stars = n+1

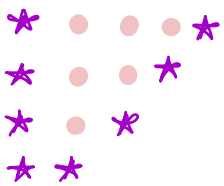
stars = n+1 − row

```
for (int row=1; row<=n; row++) {
    for (int st=1; st<=(n+1-row); st++){
        sop("*");
    }
    sopln();
}
```

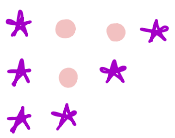
Pattern with spaces

8) Given N as input, print the following pattern.

n = 4



n = 3



Sherlocks table → according to row spaces many

[n = 4]

→ we print ★ in beginning and end

row		spaces		N
1	+	3	=	4
2	+	2	=	4
3	+	1	=	4
4	+	0	=	4

row + spaces = N

spaces = N − row

```
for ( int row=1 ; row <=N;  row ++) {
        Sop (" * ");
        for (int sp=1;  sp <= N-row ;  sp++){
                Sop (" ");
        }
        Sop ("*");
        Sopln ();
}
```

---

9) Given N as input, print the following output

N = 4

N = 3

Sherlocks table (n=4)

row no	stars	spaces
1	1	3
2	2	2
3	3	1
4	4	0

① Row = stars

② space ? row no

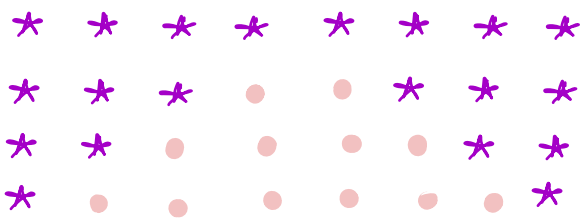
row no + space = n

space = n − row no

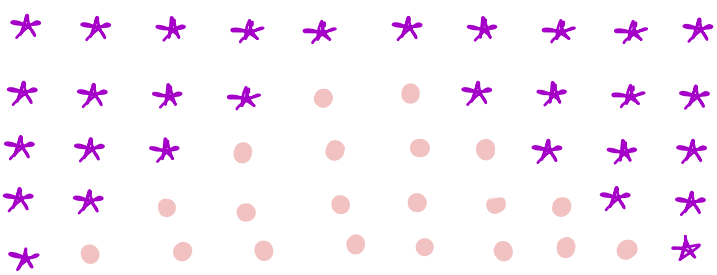
```
for (int row=1 ; row <=n ; row++) {
    for (int sp=1; sp <= (n-row); sp++) {
        sop(" ");
    }
    for (int st= 1; st <= row ; st++) {
        sop ("*");
    }
    sopln ();
}
```

9) Hollow pyramid

n = 4



n = 5



Shortcuts table [n = 4]

row	st 1	st 2
1	4	4
2	3	3
3	2	2
4	1	1

st 1 = n - row + 1

2*row	row	space
2	1	0
4	2	2
6	3	4
8	4	6

(row: +1 ↓ ; space: +2 ↓)

space	2*row
0	2
2	4
4	6
6	8

Space = 2*row - 2

3 sections

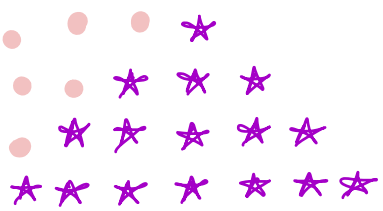
star1 = n - row + 1

spaces = 2 * row - 2

star2 = n - row + 1

Pyramid

Input = 4



Input = 3

row , space , star ?

Sherlocks table

[n=4]

2 * row	row	space	star
2	1	3	1
4	2	2	3
6	3	1	5
8	4	0	7

row: +1, +1

star: +2, +2

row + space = n

space = n - row

2 * row = star + 1

star = 2 * row - 1