⭐ STRONG YOUR LOGIC BUILDING ⭐

WITH STAR PATTERNS

Solve these **25 questions** to master your **logic and thinking ability.**

It will help you **think faster** and **build better logic** for DSA questions.

=>Print the following Patterns:

1. Print a Single Star (*)

Code:

```
System.out.println("*");
```

Explanation:
We only have to print **one single** *, so **no need for loops** here.

---

2. Print Four Stars (****)

Code:

```
System.out.println("****");
```

Explanation:
Again, just a simple **single print statement** — no loops needed!

---

3. Print n Stars on Same Line

✅ Code:

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

Explanation:

- `for (int i = 0; i < n; i++)` → **Loop** from `i = 0` to `i = n-1`.
- Inside the loop, **print** `*` **without newline** using `System.out.print("*");`.
- So all stars come in **the same line**.

---

4. Print Square of Stars (n x n Stars)

Code:

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

Code:

```java
for (int i = 0; i < n; i++) {
    // Inner loop for columns (stars)
    for (int j = 0; j < n; j++) {
        System.out.print("*");
    }
    // Move to next line after printing each row
    System.out.println();
}
```

Explanation:

- **Outer loop** → Runs `n` times to print `n` rows.
- **Inner loop** → In each row, print `n` stars.
- After each row, do `System.out.println();` to **move to the next line**.

---

5. Print an Increasing Triangle of Stars

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

code:

```
for (int i = 1; i <= n; i++) {
    // Print stars i times
    for (int j = 0; j < i; j++) {
        System.out.print("*");
    }
    // Move to next line after each row
    System.out.println();
}
```

Explanation:

- Outer loop runs from `i = 1` to `i = n`.
- In each row, **print `i` stars**.
- After printing all stars in a row, **move to the next line** using `System.out.println();`.

---

6. Print a Right-Aligned Triangle of Stars

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

Code:

```
for (int i = 0; i < n; i++) {
    // Print spaces
    for (int j = 0; j < n - i - 1; j++) {
        System.out.print(" ");
    }
    // Print stars
    for (int k = 0; k <= i; k++) {
        System.out.print("*");
    }
    // Move to next line after each row
    System.out.println();
}
```

Explanation:

- **First inner loop:** Print `(n-i-1)` spaces.
- **Second inner loop:** Print `(i+1)` stars after spaces.
- After printing spaces and stars for one row, use `System.out.println();` to **move to the next line**.

---

7. Print Stars in Even Numbers (2, 4, 6, 8, 10)

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

Code:

```java
for (int i = 1; i <= n; i++) {
    // Print stars (2 * i times)
    for (int j = 0; j < 2 * i; j++) {
        System.out.print("*");
    }
    // Move to next line after each row
    System.out.println();
}
```

Explanation:

- Outer loop runs from `i = 1` to `i = n`.
- In each row, **print `2*i` stars**.
- After printing stars for a row, **move to the next line** with `System.out.println();`.

---

8. Print Stars in Odd Numbers (1, 3, 5, 7, 9)

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

Code:

```
for (int i = 0; i < n; i++) {
    // Print stars (2*i + 1) times
    for (int j = 0; j < 2 * i + 1; j++) {
        System.out.print("*");
    }
    // Move to next line after each row
    System.out.println();
}
```

Explanation:

- Outer loop runs from `i = 0` to `i = n-1`.
- For each row:
    - Number of stars = **`2*i + 1`** (always an odd number).
    - Think like for every i we need 2*i+1
- After printing stars for the row, **move to the next line** with `System.out.println();`.

---

9. Print a Centered Pyramid of Stars

Pattern:

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

```
for (int i = 0; i < n; i++) {
    // Print spaces
    for (int j = 0; j < n - i - 1; j++) {
        System.out.print(" ");
    }
    // Print stars (2*i + 1) times
    for (int j = 0; j < 2 * i + 1; j++) {
        System.out.print("*");
    }
    // Move to next line after each row
    System.out.println();
}
```

🧠 **Explanation:**

- Outer loop runs from `i = 0` to `i = n-1`.
- **First inner loop:** Print `(n-i-1)` spaces to **shift stars to the center**.
- **Second inner loop:** Print `(2*i + 1)` stars (1, 3, 5, 7, 9...).
- After printing spaces + stars for one row, **move to the next line** with `System.out.println();`.

---

10. Print Stars and Spaces Alternating (Stars and Blank Spaces)

I have added b instead of space . Think like b is your blank space

```
for (int i = 0; i < n; i++) {
    // Print spaces (b)
    for (int j = 0; j < n - i - 1; j++) {
        System.out.print("b");
    }
    // Print stars and alternate spaces
    for (int j = 0; j <= i; j++) {
        if (j % 2 == 0) {
            System.out.print("*");
        } else {
            System.out.print("b");
        }
    }
    // Move to next line after each row
    System.out.println();
}
```

Explanation:

- **Outer loop** runs from `i = 0` to `i = n-1`.
- **First inner loop:** Print `(n-i-1)` spaces (`b`) to align stars correctly.
- **Second inner loop:** Alternate between printing stars (`*`) and blank spaces (`b`) using `if (j % 2 == 0)` to check for even and odd positions.
- After printing stars and spaces for one row, **move to the next line** with `System.out.println();`.

---

11. Print Numbers in an Increasing Sequence (1, 12, 123, 1234, 12345)

```
1
12
123
1234
12345
```

```java
for (int i = 1; i <= n; i++) {
    // Print numbers from 1 to i
    for (int j = 1; j <= i; j++) {
        System.out.print(j);
    }
    // Move to the next line after each row
    System.out.println();
}
```

Explanation:

- **Outer loop** runs from `i = 1` to `i = n`.
- **Inner loop** prints numbers starting from `1` up to `i` in each row.
- After printing the numbers in each row, **move to the next line** with `System.out.println();`.

---

12. Print Repeated Numbers per Row (Same Number Repeated)

```
1
22
333
4444
55555
```

Code:

```java
public class NumberPatternPrinter {
    public static void main(String[] args) {
        int rows = 5; // Number of rows we want

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

We have **two `for` loops**:

- **Outer loop** runs for each **row** (from 1 to 5).
- **Inner loop** prints the **current row number (`i`) exactly `i` times**.
- After printing for one row, we move to the **next line** using `System.out.println()`.

13.

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

Code:

```
public class NumberPatternPrinter {
    public static void main(String[] args) {
        int rows = 5;  // Total rows
        int num = 1;   // Starting number

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

explanation:

- Start from `num = 1`.
- For each **row `i`**, print `i` numbers.
- After each number, **add a space**.
- After each row, **go to the next line**.
- Keep **incrementing** `num` after printing.

14.

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

```
int rows = 7;
int num = 1;

for (int i = 1; i <= rows; i++) {
    for (int j = 1; j <= i; j++) {
        System.out.print((num % 10) + " ");
        num++;
    }
    System.out.println();
}
```

Explanation:

- `num` starts at `1` and increases after every print.
- `num % 10` keeps only the last digit (wraps after 9 to 0).
- Outer loop runs for each row.
- Inner loop prints `i` numbers in row `i`.
- After each row, move to the next line.

15.

```
1
0 1
0 1 0
1 0 1 0
1 0 1 0 1
```

Explanation:

- Loop runs for 5 rows.
- Inner loop prints `i` values in row `i`.
- The value to print is `(i + j) % 2`:
    - This alternates between 0 and 1 based on the sum of row and column indexes.
- The pattern starts with `1` and alternates accordingly.

```java
int rows = 5;

for (int i = 1; i <= rows; i++) {
    for (int j = 1; j <= i; j++) {
        int val = (i + j) % 2;
        System.out.print(val + " ");
    }
    System.out.println();
}
```

16

```
A
B C
D E F
G H I J
K L M N O
```

```java
int rows = 5;
char ch = 'A';

for (int i = 1; i <= rows; i++) {
    for (int j = 1; j <= i; j++) {
        System.out.print(ch + " ");
        ch++;
    }
    System.out.println();
}
```

Explanation:

- `char ch = 'A'` initializes the starting alphabet.
- Outer loop runs from 1 to 5 (for 5 rows).
- Inner loop runs `i` times for row `i`.
- Each character is printed and then incremented (`ch++`).
- Output continues alphabetically from `A` to `O`.

17

```
A
B B
C C C
D D D D
E E E E E
```

Code:

```
int rows = 5;

for (int i = 0; i < rows; i++) {
    char ch = (char) ('A' + i);
    for (int j = 0; j <= i; j++) {
        System.out.print(ch + " ");
    }
    System.out.println();
}
```

Explanation:

- `rows = 5` → pattern has 5 rows.
- Outer loop runs from `0` to `4`.
- In each row, calculate the character as `'A' + i`:
    - Row 0 → A, Row 1 → B, ..., Row 4 → E.
- Inner loop prints the same character `i+1` times.
- `System.out.println()` moves to next line.

18.

```
A
A B
A B C
A B C D
A B C D E
```

Code:

```
int rows = 5;

for (int i = 1; i <= rows; i++) {
    for (int j = 0; j < i; j++) {
        char ch = (char) ('A' + j);
        System.out.print(ch + " ");
    }
    System.out.println();
}
```

Explanation:

- `rows = 5` → 5 rows in the pattern.
- Outer loop runs from `1` to `5` (row count).
- Inner loop runs `j < i` to print increasing letters in each row.
- `'A' + j` gives the next character (A, B, C, ...).
- Characters always start from `'A'` in each row.
- `System.out.println()` moves to the next line.

19.

```
    A
   BCD
  EFGHI
 JKLMNOP
QRSTUVWXY
```

Code:

```java
int rows = 5;
char ch = 'A';

for (int i = 1; i <= rows; i++) {
    // Print leading spaces for alignment
    for (int j = i; j < rows; j++) {
        System.out.print(" ");
    }

    // Print the characters in each row
    for (int j = 1; j <= (2 * i - 1); j++) {
        System.out.print(ch);
        ch++;
    }

    System.out.println();  // Move to next line
}
```

Explanation:

- `rows = 5`: Defines the number of rows.
- `char ch = 'A'`: Start from character 'A'.
- Outer loop (`i`) runs for each row.
- **Leading spaces**: Inner loop prints spaces to align the characters in a pyramid shape.
- **Print characters**: The second inner loop prints `2 * i - 1` characters for row `i` (odd number of characters).
- After printing, increment the character (`ch++`) and move to the next line using `System.out.println()`.

20.

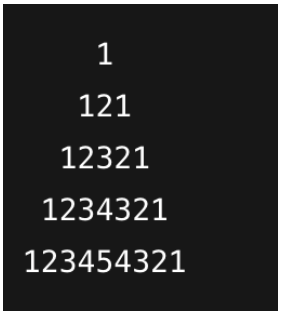
```
1
12
123
1234
12345
```

```java
int rows = 5;

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

🧠 Explanation:

- `rows = 5`: Defines the number of rows.
- Outer loop (`i`) runs from `1` to `5` (for 5 rows).
- Inner loop (`j`) prints numbers from `1` to `i` for each row.
- `System.out.println()` moves to the next line after printing each row.

21.




```java
int rows = 5;

for (int i = 1; i <= rows; i++) {
    // Print leading spaces for alignment
    for (int j = i; j < rows; j++) {
        System.out.print(" ");
    }

    // Print ascending numbers
    for (int j = 1; j <= i; j++) {
        System.out.print(j);
    }

    // Print descending numbers
    for (int j = i - 1; j >= 1; j--) {
        System.out.print(j);
    }

    System.out.println();  // Move to next line
}
```

Explanation:

- `rows = 5`: Defines the number of rows.
- Outer loop (`i`) runs from `1` to `5` (for 5 rows).
- **Leading spaces**: Inner loop prints spaces for alignment to form the pyramid shape.
- **Ascending numbers**: Print numbers from `1` to `i`.
- **Descending numbers**: Print numbers from `i-1` down to `1`.
- `System.out.println()` moves to the next line after each row.

22.

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

```java
int rows = 5;

// Upper half of the pattern
for (int i = 1; i <= rows; i++) {
    for (int j = 1; j <= i; j++) {
        System.out.print("*");
    }
    System.out.println();
}

// Lower half of the pattern
for (int i = rows - 1; i >= 1; i--) {
    for (int j = 1; j <= i; j++) {
        System.out.print("*");
    }
    System.out.println();
}
```

Explanation:

- `rows = 5`: Defines the number of rows for the upper half of the pattern.
- The first loop prints the upper part of the pattern, starting from 1 star up to `rows` stars.
- The second loop prints the lower half of the pattern, starting from `rows-1` stars down to 1 star.
- `System.out.println()` moves to the next line after printing each row.

23.

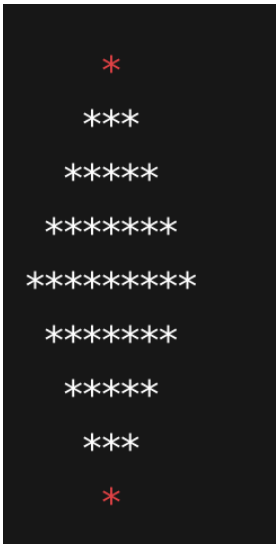
```
*
**
***
****
*****
*****
****
***
**
*
```

```java
int rows = 5;

// Upper half of the pattern
for (int i = 1; i <= rows; i++) {
    for (int j = 1; j <= i; j++) {
        System.out.print("*");
    }
    System.out.println();
}

// Lower half of the pattern (starting from the same number of stars as the middle row)
for (int i = rows; i >= 1; i--) {
    for (int j = 1; j <= i; j++) {
        System.out.print("*");
    }
    System.out.println();
}
```

Explanation:

- **`rows = 5`**: Defines the number of rows for the first half of the pattern.
- The first loop prints the upper part of the pattern, from 1 star to `rows` stars.
- The second loop prints the lower half, starting from `rows` stars down to 1 star.
- `System.out.println()` moves to the next line after printing each row.

24



Code:

```java
int rows = 5;

// Upper half of the pattern (including middle row)
for (int i = 1; i <= rows; i++) {
    // Print leading spaces
    for (int j = i; j < rows; j++) {
        System.out.print(" ");
    }
    // Print stars
    for (int j = 1; j <= (2 * i - 1); j++) {
        System.out.print("*");
    }
    System.out.println();
}

// Lower half of the pattern
for (int i = rows - 1; i >= 1; i--) {
    // Print leading spaces
    for (int j = rows; j > i; j--) {
        System.out.print(" ");
    }
    // Print stars
    for (int j = 1; j <= (2 * i - 1); j++) {
        System.out.print("*");
    }
    System.out.println();
}
```

Explanation:

- `rows = 5`: Defines the number of rows for the upper part of the pattern (excluding the middle row).
- The first loop prints the upper half of the pattern:
  - For each row, print the leading spaces first, then the stars (`2 * i - 1` stars for row `i`).
- The second loop prints the lower half of the pattern:
  - Similar to the first loop but starts from `rows - 1` and prints fewer stars as the row number decreases.
- `System.out.println()` moves to the next line after printing each row.

25.

```
    5
   545
  54345
 5432345
543212345
```

```java
int rows = 5;

for (int i = 1; i <= rows; i++) {
    // Print leading spaces
    for (int j = i; j < rows; j++) {
        System.out.print(" ");
    }

    // Print descending numbers from 5
    for (int j = 5; j > 5 - i; j--) {
        System.out.print(j);
    }

    // Print ascending numbers from i
    for (int j = 5 - i + 1; j <= 5; j++) {
        System.out.print(j);
    }

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

Explanation:

- `rows = 5`: Defines the number of rows.
- The first loop prints the leading spaces to center-align the pattern.
- The second loop prints numbers in descending order, starting from `5` and decreasing until the appropriate number for each row.
- The third loop prints numbers in ascending order, starting from the number after the descending ones.
- `System.out.println()` moves to the next line after each row.

SHOW SOME LOVE BY FOLLOWING CodeWithNishchal ❤️