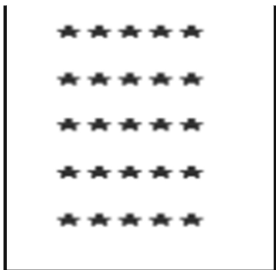
Pattern-1: Rectangular Star Pattern

Problem Statement: Given an integer **N,** print the following pattern.



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

Examples:

```
Example 1:
Input: N = 3
Output:
* * *
* * *
* * *


Example 2:
Input: N = 6
Output:
* * * * * *
* * * * * *
* * * * * *
* * * * * *
* * * * * *
* * * * * *
```

Solution

Disclaimer: Don't jump directly to the solution, try it out yourself first.

Problem Link

Approach:

There are 4 general rules for solving a pattern-based question:

- We always use nested loops for printing the patterns. For the outer loop, we count the number of lines/rows and loop for them.
- Next, for the inner loop, we focus on the number of columns and somehow connect them to the rows by forming a logic such that for each row we get the required number of columns to be printed.
- We print the '*' inside the inner loop.
- Observe symmetry in the pattern or check if a pattern is a combination of two or more similar patterns.

In this particular problem, we run the outer loop for N times since we have N rows to be printed, the inner loop also runs for N times as we have to print N stars in each row. This way we get a rectangular star pattern (square) with an equal number of rows and columns.

Code:

C++Java

```
#include <bits/stdc++.h>
using namespace std;

void pattern1(int N)
{
    // This is the outer loop which will loop for the rows.
    for (int i = 0; i < N; i++)
    {
        // This is the inner loop which here, loops for the columns
        // as we have to print a rectangular pattern.
        for (int j = 0; j < N; j++)
        {
            cout << "* ";
        }

        // As soon as N stars are printed, we move to the
        // next row and give a line break otherwise all stars
        // would get printed in 1 line.
        cout << endl;
    }
}

int main()
{
    // Here, we have taken the value of N as 5.
    // We can also take input from the user.
    int N = 5;

    pattern1(N);

    return 0;
}
```

Output

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