Assignment: Loop-Based Pattern Problems

Assignment: Loop-Based Pattern Problems

Question 1: Right-Angled Triangle Pattern (Sequential Numbers)

Problem Statement:
Print a right-angled triangle pattern using integers. Each row contains sequential numbers starting from 1.

Input Format:
- First line of input contains a single integer **N** - the size of the triangle.

Constraints:
- 1 <= N <= 50

Output Format:
- Print the right-angled triangle pattern.

Example:
Input:
6

Output:
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
16 17 18 19 20 21

Question 2: Right-Angled Triangle Pattern (Column Incrementing)

Problem Statement:
Print a right-angled triangle pattern where each column increments separately.

Input Format:
- First line of input contains a single integer **N** - the size of the triangle.

Constraints:
- 1 <= N <= 50

Output Format:

- Print the right-angled triangle pattern.

Example:
Input:
5

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

Question 3: Hollow Rectangle Pattern

Problem Statement:
Print a hollow rectangle pattern using '*' characters.

Input Format:
- Two integers **W** (width) and **L** (length) of the rectangle.

Constraints:
- 2 <= W <= 50
- 2 <= L <= 50

Output Format:
- Print the hollow rectangle pattern.

Example:
Input:
5 4

Output:
*****
*   *
*   *
*****

Question 4: Half Diamond Pattern

Problem Statement:

Print a half diamond pattern using '*' characters.

Input Format:
- A single integer **N**.

Constraints:
- 1 <= N <= 50

Output Format:
- Print the half diamond pattern.

Example:
Input:
5

Output:
*
**
***
****
*****
****
***
**
*

Question 5: Hollow Half Inverted Pyramid

Problem Statement:
Print a hollow half inverted pyramid pattern using '*' characters.

Input Format:
- A single integer **N** - the size of the pyramid.

Constraints:
- 1 <= N <= 50

Output Format:
- Print the hollow half inverted pyramid pattern.

Example:
Input:
5

Output:

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

Instructions:

1. Solve each problem using loops in a programming language of your choice.
2. Ensure that the programs correctly handle edge cases.
3. Submit your solution along with test cases verifying different inputs.
4. Maintain proper code indentation and readability.