ISLAMIC UNIVERSITY OF TECHNOLOGY (IUT)

ORGANISATION OF ISLAMIC COOPERATION (OIC)

Department of Computer Science and Engineering (CSE)

CSE 4108: Structured Programming I Lab
Lab 7, Section 2A

Objectives

- Nested Loops
- Printing Patterns

Guidelines

- All source codes (.c files) in the naming format `ID_Lab7_TaskN.c`, eg. `230041101_Lab7_Task1.c`.
- Ensure that your program produces the correct output for all sample cases.
- Source codes must be properly indented
- Screenshots are not required.
- Throughout the rest of the semester, try to regularly solve problems and participate in contests on online judges such as *Codeforces*, *Codechef*, *Atcoder*.

Task 1 — Learning to Add

The ICPC (International Collegiate Programming Contest) is just around the corner, and the team is busy preparing for the big event. The Preliminary Contest for ICPC Dhaka Regional Contest, hosted by Daffodil International University, will take place in November.

As the ICPC (International Collegiate Programming Contest) approaches, the team at IUT is practicing their arithmetic skills. They find that understanding the cumulative sum can help in a variety of competitive programming scenarios.

Your task is to write a C program that takes an integer n as input and prints the cumulative sum of natural numbers from 1 up to n in the form of direct addition on separate lines.

Sample Execution(s)

Input 1

```
Enter a number: 3
```

Output 1

```
1 = 1
1 + 2 = 3
1 + 2 + 3 = 6
```

Input 2

```
Enter a number: 5
```

Output 2

```
1 = 1
1 + 2 = 3
1 + 2 + 3 = 6
1 + 2 + 3 + 4 = 10
1 + 2 + 3 + 4 + 5 = 15
```

Task 2 — Inverted Pyramid of Numbers

As the ICPC approaches, the team at IUT is eager to enhance their pattern recognition skills. They decide to practice printing patterns.

Your task is to write a C program that takes an integer input, representing the number of rows, and prints an inverted pyramid of numbers. The pattern should start with a full row of numbers from 1 to n and decrease by one number in each subsequent row, with appropriate indentation. See the sample output for more clarity.

Sample Execution(s)

Input 1

```
Enter the number of rows: 5
```

Output 1

```
12345
 1234
  123
   12
    1
```

Input 2

```
Enter the number of rows: 3
```

Output 2

```
123
 12
  1
```

Input 3

```
Enter the number of rows: 4
```

Output 3

```
1234
 123
  12
   1
```

Task 3 — Running out of Time

Time is of the essence in ICPC, and your team finds that the clocks have stopped working in the contest arena. Left with no other options, you decide to make your own hourglass to keep track of time.

To prepare for this unexpected twist, your mission is to write a C program that takes an integer as input and prints an hourglass pattern using asterisks (‘*‘). The hourglass will start with a full row of asterisks and then decrease in width with each subsequent row until it reaches one asterisk, after which it will symmetrically increase back to the full width.

Your task is to ensure that the hourglass is well-aligned and clearly defined, providing a visual representation of this unique challenge.

Sample Execution(s)

Input 1

```
Enter the number of rows: 3
```

Output 1

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

Input 2

```
Enter the number of rows: 2
```

Output 2

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

Task 4 — Arcane Tangent

As the team prepares for the ICPC World Finals, the pressure is palpable. Competing against the brightest minds in the world, every calculation must be flawless. This year's hardest problem depends on the arctan function - it can mean the difference between victory and defeat.

To succeed, the team must harness the power of mathematical approximations. They decide to focus on accurately computing $tan^{-1}(x)$ using its Taylor series expansion, as even the smallest error could lead to catastrophic results.

In this challenge, your task is to help the team approximate $\tan^{-1}(x)$ for a given x using the series:

$$\tan^{-1}(x) \approx x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \cdots$$

The program should take an input value x (where $|x| \leq 1$) and the number of terms to use in the series for the approximation.

Sample Execution(s)

Input 1

```
Enter the value of x: 0.5
Enter the number of terms to use: 5
```

Output 1

```
Approximate value of tan^-1(0.5) = 0.463684
```

Input 2

```
Enter the value of x: 1.0
Enter the number of terms to use: 7
```

Output 2

```
Approximate value of tan^-1(1.0) = 0.820935
```