

**Department of Computer and Communication Engineering (CCE)**

**LAB REPORT**

**Course Title :** **Competive Programming Sessional**

**Course Code : CCE-2310**

**Session Topic : Multiplicative Triangle**

**Submitted By**

**Name : Mossammat Naznin Akter Riya**

**ID No : E233414**

**Semester : 3rd**

**Section : 3AF**

**Date of Experiment: - -2024**

**Date of Submission: 08-12-2024**

**Submitted To**

**Remarks**

**Sukanta Paul**

**Adjunct faculty, Department of CCE, IIUC**



**Experiment No: 10**

**Experiment Name:** Multiplicative Triangle

**Problem Description:** Suppose, value of n is provided to you. Then you have to print a multiplicative triangle of n .if n=3 there will be 3 lines and first line will contain 1 value, 2nd line contains 2 value and 3rd line contains 3 value and so on. The value printed on each line is multiplication table of n.

**Input Format**

So, in the input format first line contains the value of test cases t.following t lines contains the value of n.

**Constraints**

n>0 and n<11

**Output Format**

Show the output as given in output format.

**Objectives:**

1. To understand and implement the concept of multiplicative triangles in programming.
2. To practice using nested loops for generating structured outputs dynamically.
3. To efficiently handle multiple test cases with clear input and output formatting.
4. To enhance problem-solving skills by integrating mathematical logic into programming.

**Explanation after analysis:** The program generates a multiplicative triangle based on the input value n, where each line displays the multiplication results of n up to the respective line number. By utilizing nested loops, the program calculates the products dynamically, ensuring efficiency and avoiding redundant storage. The constraints, with n<11n < 11n<11, ensure the calculations are manageable within the given limits. The formatted output neatly separates the results of different test cases, making the program both functional and user-friendly. This implementation highlights the effective use of loops and modular programming to solve structured pattern problems.

**Source code:**

#include <stdio.h>

int main() {

int t, n;

scanf("%d", &t);

for (int test\_case = 1; test\_case <= t; test\_case++)

{

scanf("%d", &n);

printf("Test Case %d:\n", test\_case);

int counter = 1;

for (int i = 1; i <= n; i++)

{

for (int j = 1; j <= i; j++) {

printf("%d", n \* counter);

counter++;

if (j < i) {

printf(" "); }

}

printf("\n");

}

}

return 0;

}

**Procedure :**

**Understand the Problem**

* Analyze the requirements to generate a multiplicative triangle based on the given input n.
* Note that the triangle should contain n lines, with each line displaying the multiplication table of n up to the line number.

**Input Handling**

* Start by taking the number of test cases, t, as input.
* For each test case, take the integer n as input, representing the size of the multiplicative triangle.

**Logic Implementation**

* Use nested loops to generate the triangle:
  + The outer loop iterates over the line numbers (from 1 to n).
  + The inner loop calculates and prints the products for each line (from 1 to the current line number).

**Formatting the Output**

* Print the multiplication results for each line with values separated by spaces.
* Insert a newline after each line.
* Add a blank line between the outputs of different test cases for clarity.

**Execution and Testing**

* Compile and run the program with sample inputs to verify correctness.
* Test the program with edge cases (e.g., smallest and largest values of n).

**Output Results**

* For each test case, display the multiplicative triangle as per the required format. Ensure outputs for multiple test cases are clearly separated.

**Sample input:**

2

3

4

**Sample output:**

Test Case 1:

3

6 9

12 15 18

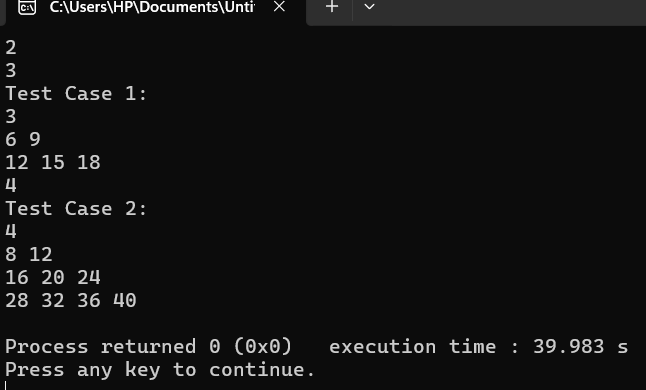
Test Case 2:

4

8 12

16 20 24

28 32 36 40

****

**Figure no: 01**

**Discussion :** The experiment demonstrates the practical application of nested loops to generate a structured pattern, specifically a multiplicative triangle, based on a given input value n. It reinforces the understanding of how loops and mathematical calculations can work together to produce dynamic outputs. The constraints ensure simplicity and manageability, while the program's ability to handle multiple test cases showcases its versatility. This task highlights the importance of logical thinking and proper formatting in programming to achieve both accuracy and clarity in the output, making it a valuable exercise in problem-solving.