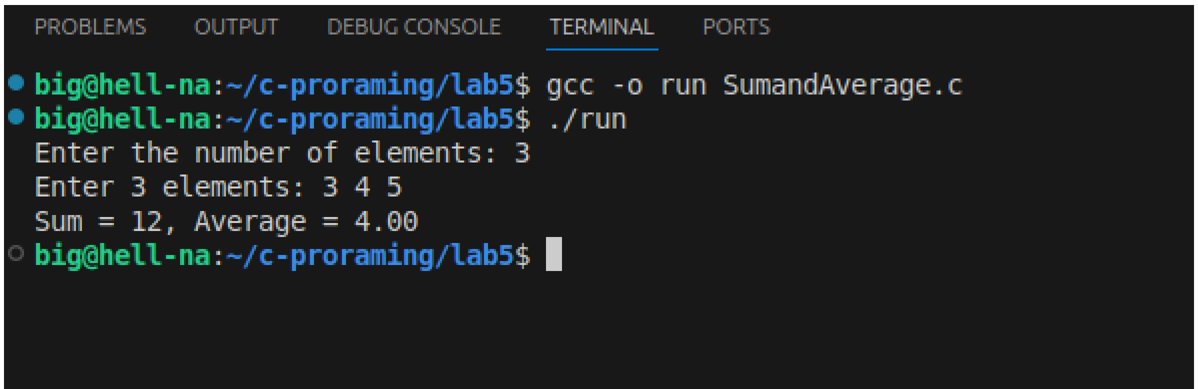
Lab – 5 (Arrays)

1. Write a C program to find the **sum and average** of elements in an array.

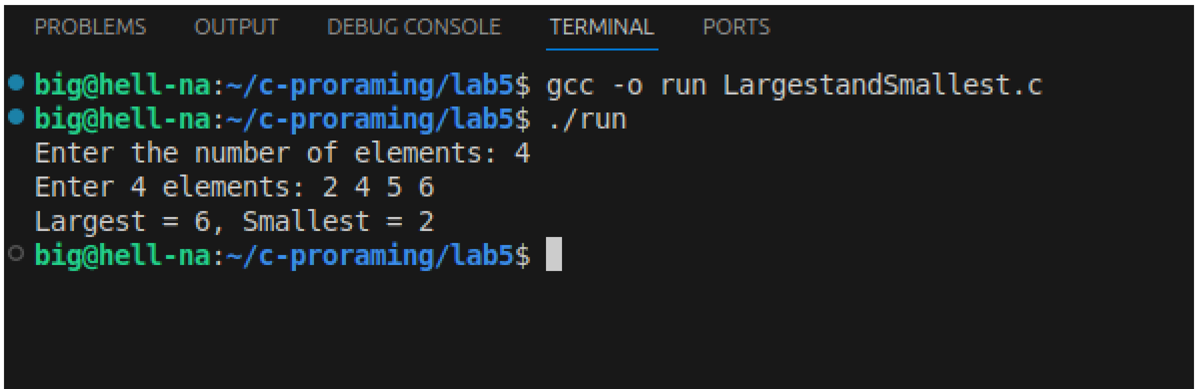
Output :



```
big@hell-na:~/c-proraming/lab5$ gcc -o run SumandAverage.c
big@hell-na:~/c-proraming/lab5$ ./run
Enter the number of elements: 3
Enter 3 elements: 3 4 5
Sum = 12, Average = 4.00
big@hell-na:~/c-proraming/lab5$
```

2. Write a C program to find the **largest and smallest** element in an array.

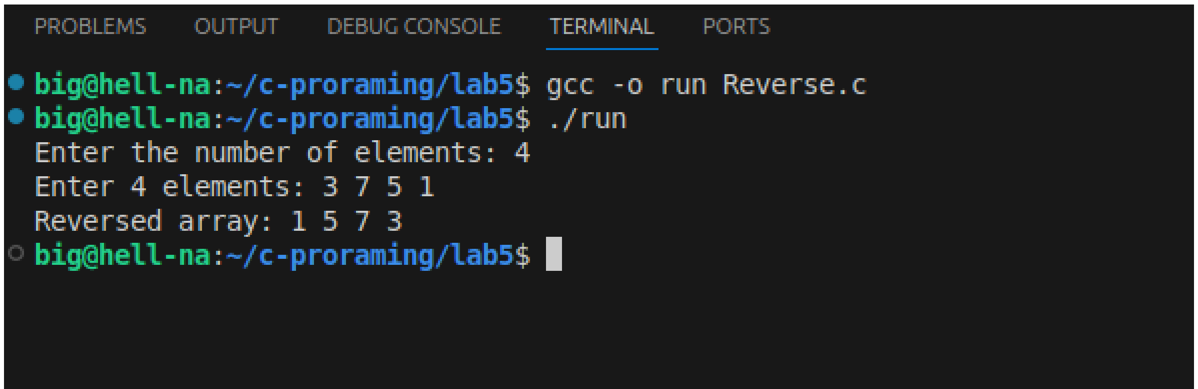
Output :


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
big@hell-na:~/c-proraming/lab5$ gcc -o run LargestandSmallest.c
big@hell-na:~/c-proraming/lab5$ ./run
Enter the number of elements: 4
Enter 4 elements: 2 4 5 6
Largest = 6, Smallest = 2
big@hell-na:~/c-proraming/lab5$ 
```

3. Write a program to **reverse an array**.

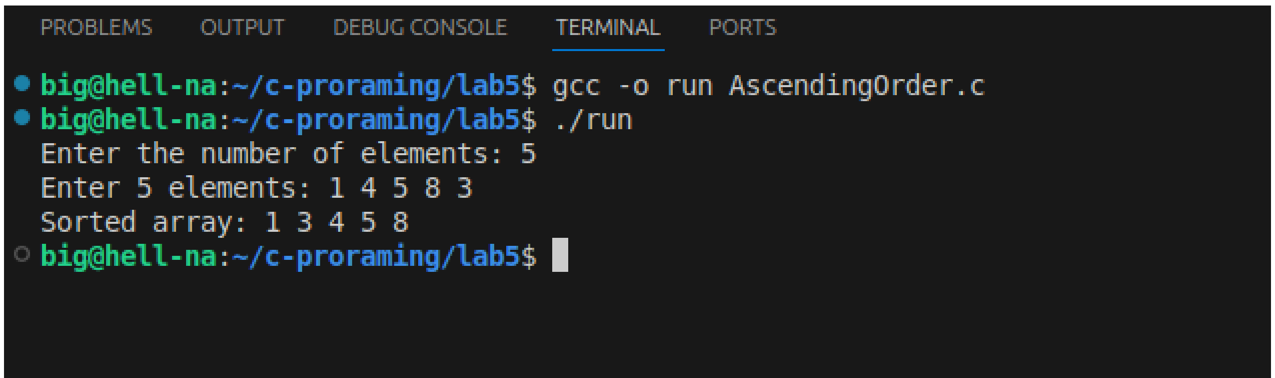
Output :



```
big@hell-na:~/c-proraming/lab5$ gcc -o run Reverse.c
big@hell-na:~/c-proraming/lab5$ ./run
Enter the number of elements: 4
Enter 4 elements: 3 7 5 1
Reversed array: 1 5 7 3
big@hell-na:~/c-proraming/lab5$
```

4. Write a C program to **sort an array in ascending order** using the Bubble Sort algorithm.

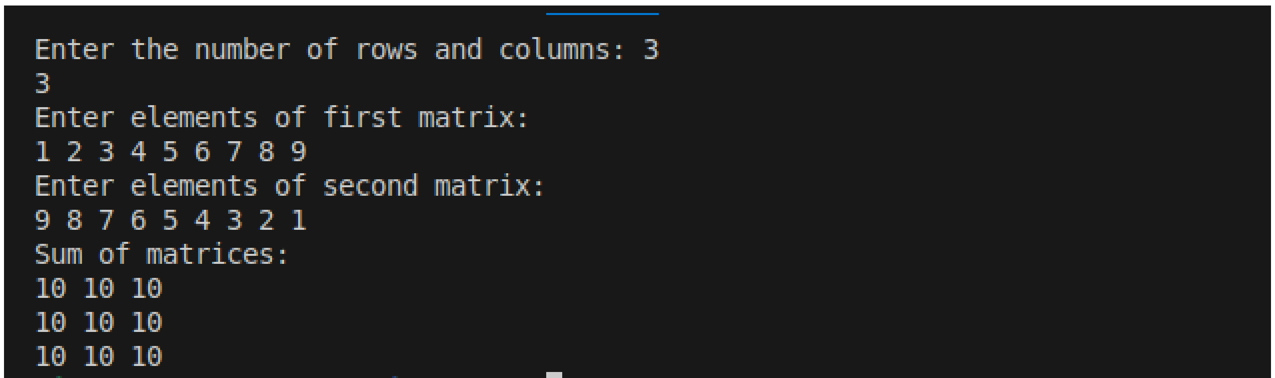
Output :



```
big@hell-na:~/c-proraming/lab5$ gcc -o run AscendingOrder.c
big@hell-na:~/c-proraming/lab5$ ./run
Enter the number of elements: 5
Enter 5 elements: 1 4 5 8 3
Sorted array: 1 3 4 5 8
big@hell-na:~/c-proraming/lab5$
```

5. Write a C program to perform **matrix addition** using a **2D array**.

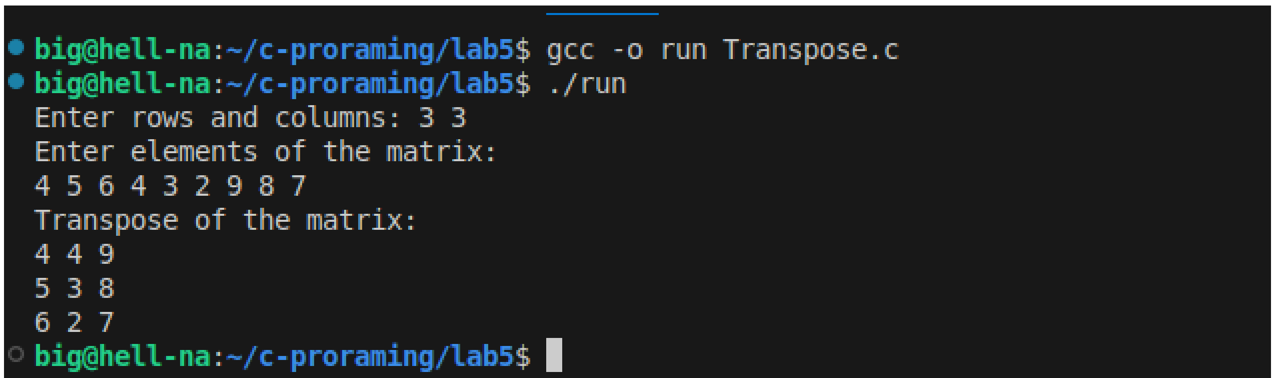
Output :



```
Enter the number of rows and columns: 3
3
Enter elements of first matrix:
1 2 3 4 5 6 7 8 9
Enter elements of second matrix:
9 8 7 6 5 4 3 2 1
Sum of matrices:
10 10 10
10 10 10
10 10 10
```

6. Write a C program to **find the transpose** of a matrix.

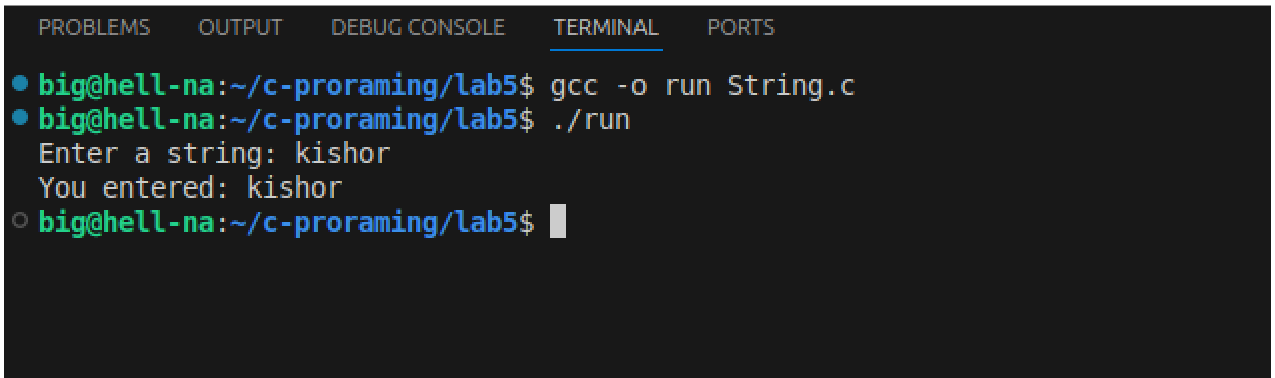
Output :



```
● big@hell-na:~/c-proraming/lab5$ gcc -o run Transpose.c
● big@hell-na:~/c-proraming/lab5$ ./run
  Enter rows and columns: 3 3
  Enter elements of the matrix:
  4 5 6 4 3 2 9 8 7
  Transpose of the matrix:
  4 4 9
  5 3 8
  6 2 7
○ big@hell-na:~/c-proraming/lab5$ 
```

7. Write a C program to **store and display a string** using a character array.

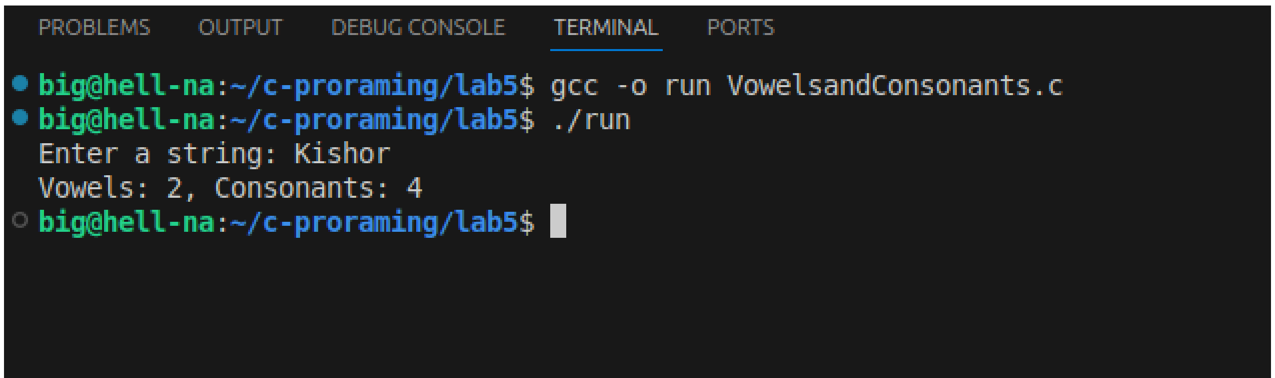
Output :



```
big@hell-na:~/c-proraming/lab5$ gcc -o run String.c
big@hell-na:~/c-proraming/lab5$ ./run
Enter a string: kishor
You entered: kishor
big@hell-na:~/c-proraming/lab5$ 
```

8. Write a C program to count **the number of vowels and consonants** in a given string.

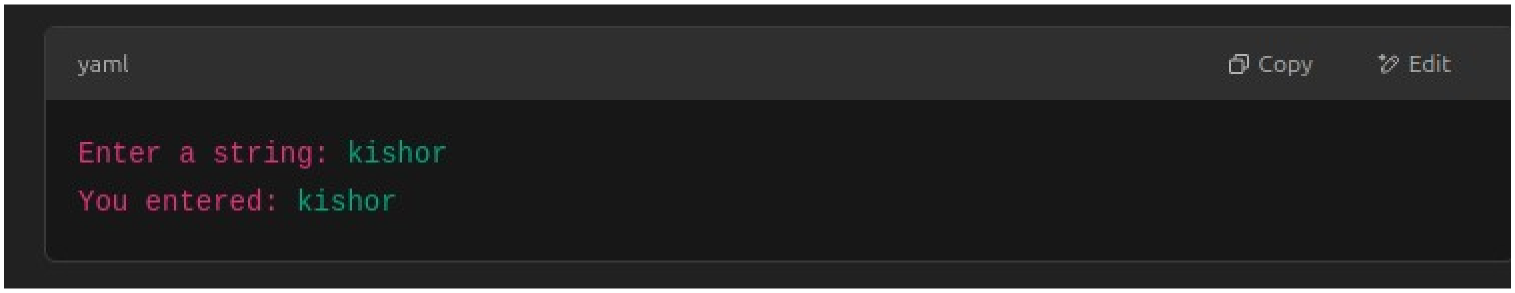
Output :



```
big@hell-na:~/c-proraming/lab5$ gcc -o run VowelsandConsonants.c
big@hell-na:~/c-proraming/lab5$ ./run
Enter a string: Kishor
Vowels: 2, Consonants: 4
big@hell-na:~/c-proraming/lab5$
```

9. Write a C program to **read a string from the user and display it** using `gets()` and `puts()`.

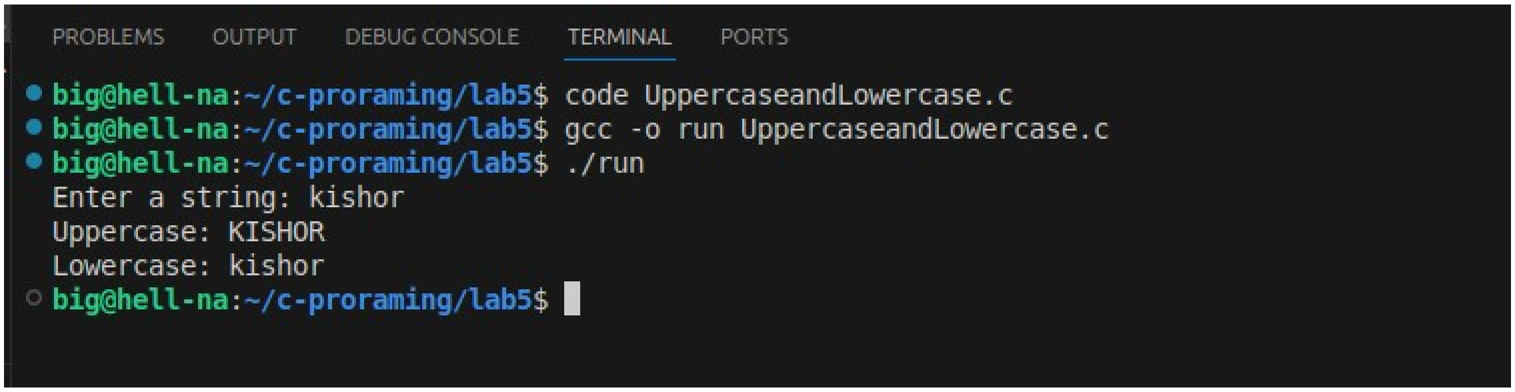
Output :



```
Enter a string: kishor
You entered: kishor
```

10.Write a program to **convert a given string to uppercase and lowercase**.

Output :

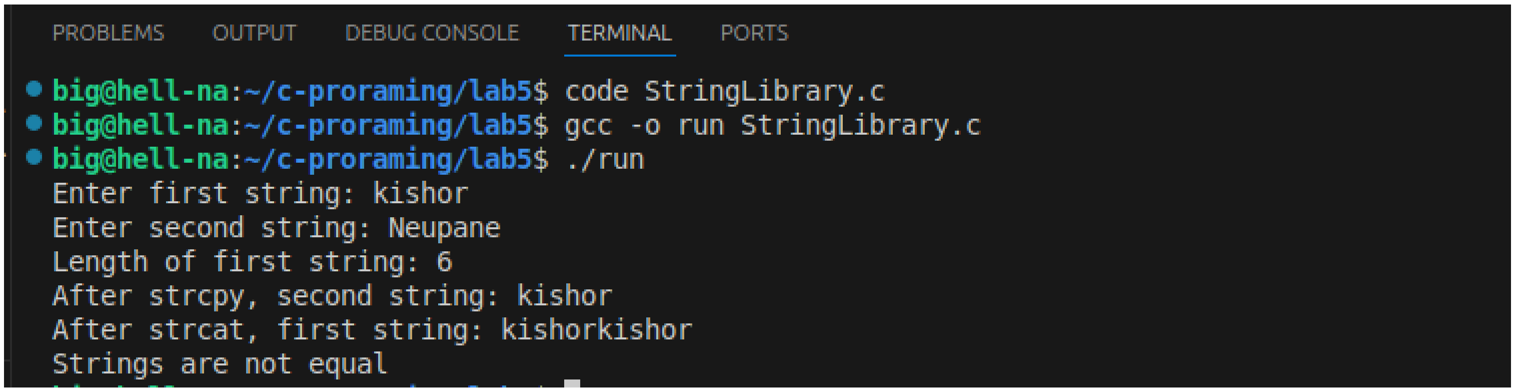


```
big@hell-na:~/c-proraming/lab5$ code UppercaseandLowercase.c
big@hell-na:~/c-proraming/lab5$ gcc -o run UppercaseandLowercase.c
big@hell-na:~/c-proraming/lab5$ ./run
Enter a string: kishor
Uppercase: KISHOR
Lowercase: kishor
big@hell-na:~/c-proraming/lab5$ 
```

11.Write a C program to demonstrate the use of String library functions:

a. Strlen() , Strcpy() , Strcat() , Strcmp()

Output :

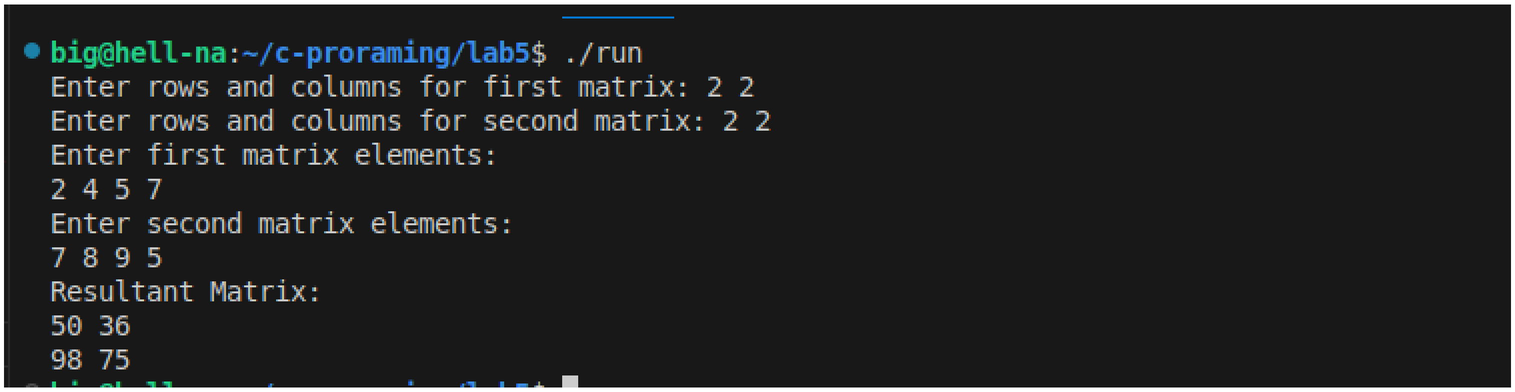


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
● big@hell-na:~/c-proraming/lab5$ code StringLibrary.c
● big@hell-na:~/c-proraming/lab5$ gcc -o run StringLibrary.c
● big@hell-na:~/c-proraming/lab5$ ./run
  Enter first string: kishor
  Enter second string: Neupane
  Length of first string: 6
  After strcpy, second string: kishor
  After strcat, first string: kishorkishor
  Strings are not equal
```

12.Write a C program to **multiply two matrices** and display the result.

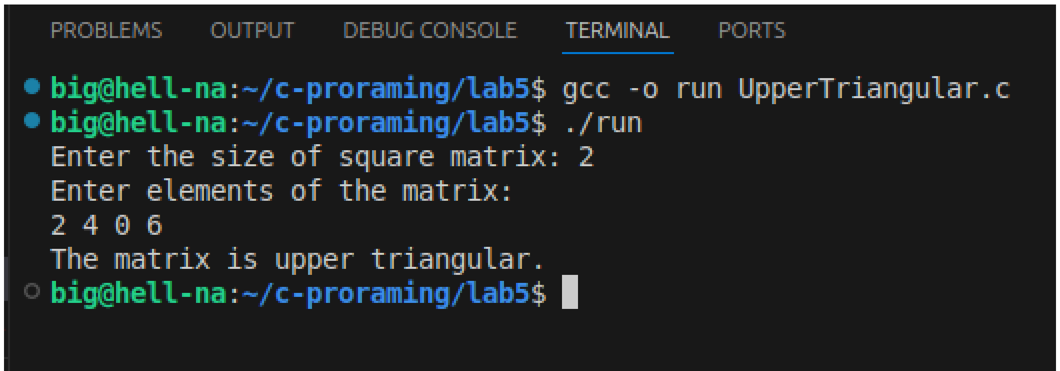
Output :



```
● big@hell-na:~/c-proraming/lab5$ ./run
  Enter rows and columns for first matrix: 2 2
  Enter rows and columns for second matrix: 2 2
  Enter first matrix elements:
  2 4 5 7
  Enter second matrix elements:
  7 8 9 5
  Resultant Matrix:
  50 36
  98 75
```

13. Write a program to check whether a **given matrix is upper triangular** (all elements below the diagonal are zero).

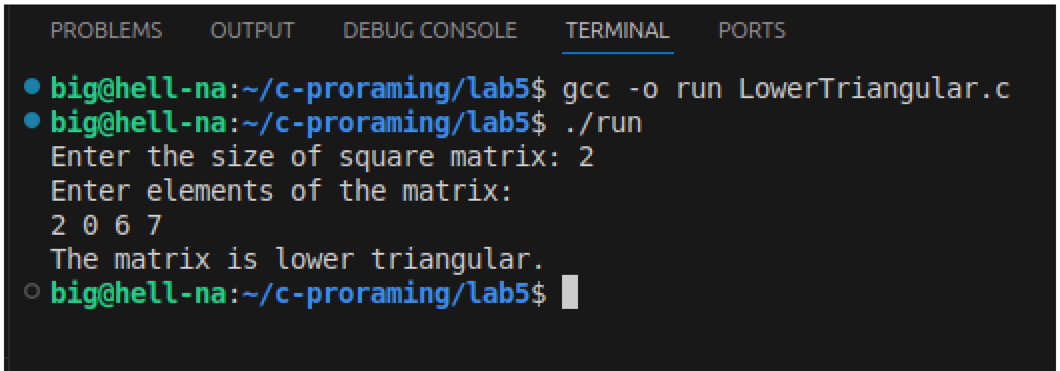
Output :



```
big@hell-na:~/c-proraming/lab5$ gcc -o run UpperTriangular.c
big@hell-na:~/c-proraming/lab5$ ./run
Enter the size of square matrix: 2
Enter elements of the matrix:
2 4 0 6
The matrix is upper triangular.
big@hell-na:~/c-proraming/lab5$
```

14. Write a program to check whether a **given matrix is lower triangular** (all elements above the diagonal are zero).

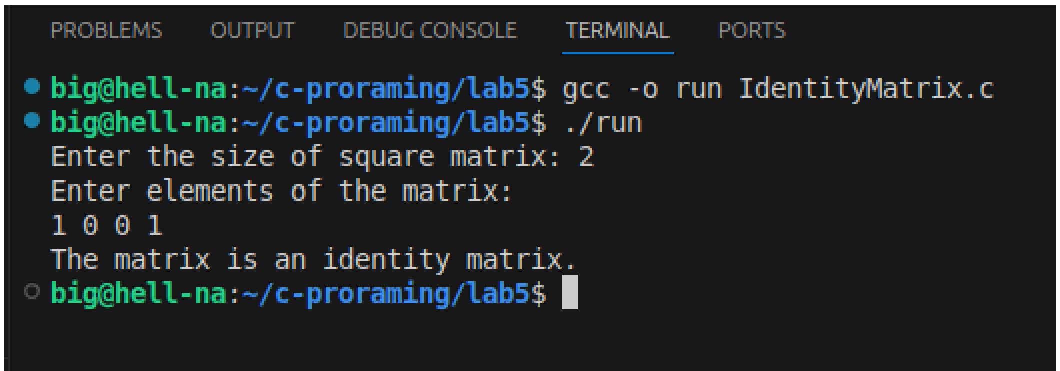
Output :



```
big@hell-na:~/c-proraming/lab5$ gcc -o run LowerTriangular.c
big@hell-na:~/c-proraming/lab5$ ./run
Enter the size of square matrix: 2
Enter elements of the matrix:
2 0 6 7
The matrix is lower triangular.
big@hell-na:~/c-proraming/lab5$
```

15. Write a program to check whether a **given matrix is an identity matrix** (all diagonal elements are 1, and other elements are 0).

Output :

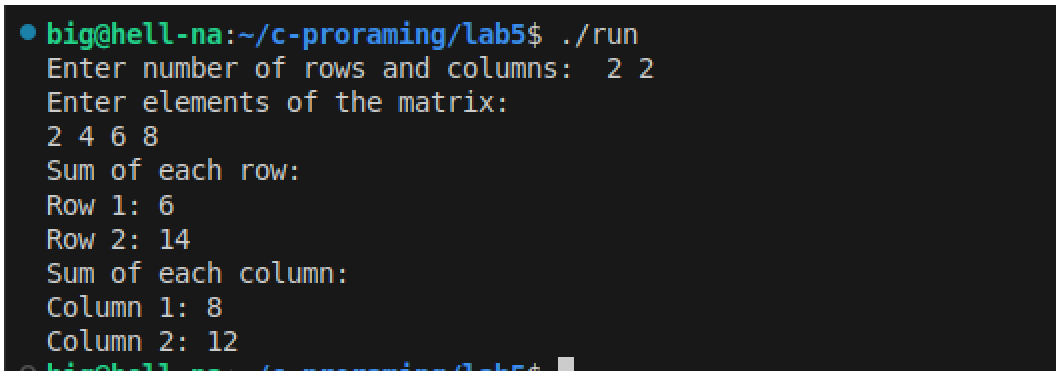


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
● big@hell-na:~/c-proraming/lab5$ gcc -o run IdentityMatrix.c
● big@hell-na:~/c-proraming/lab5$ ./run
  Enter the size of square matrix: 2
  Enter elements of the matrix:
  1 0 0 1
  The matrix is an identity matrix.
○ big@hell-na:~/c-proraming/lab5$
```

16.Write a program to **find the sum of each row and column** in a matrix.

Output :



```
big@hell-na:~/c-proraming/lab5$ ./run
Enter number of rows and columns:  2 2
Enter elements of the matrix:
2 4 6 8
Sum of each row:
Row 1: 6
Row 2: 14
Sum of each column:
Column 1: 8
Column 2: 12
```