**Batch: C4-1 Roll No.: 16010123217**

**Experiment / assignment / tutorial No. 8**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **TITLE:**  Write a program in C to demonstrate use of a pointer. |

**AIM:** 1) Write a program that calculates and prints the transpose of a given matrix using pointers.

2) Write a file copy program in C that copies a file into another.

\_\_\_\_\_\_**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected OUTCOME of Experiment:**

Apply concepts of pointers in dynamic memory allocation and file handling(CO5).

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

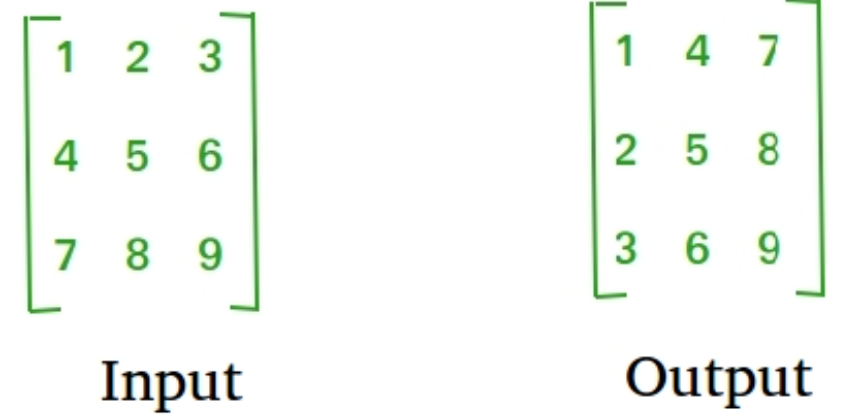
**Books/ Journals/ Websites referred:**

1. Programming in C, second edition, Pradeep Dey and Manas Ghosh, Oxford University Press.
2. Programming in ANSI C, fifth edition, E Balagurusamy, Tata McGraw Hill.
3. Introduction to programming and problem solving , G. Michael Schneider ,Wiley India edition.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Problem Definition:**

1. The program allows the user to input a matrix, dynamically allocates memory for the matrix and its transpose, calculates and prints the transpose of the matrix using pointers, and then frees the dynamically allocated memory. For example



1. The program copies the contents of a source file to the destination file, character by character.

**Algorithm:**

**1) Transpose of a Matrix**

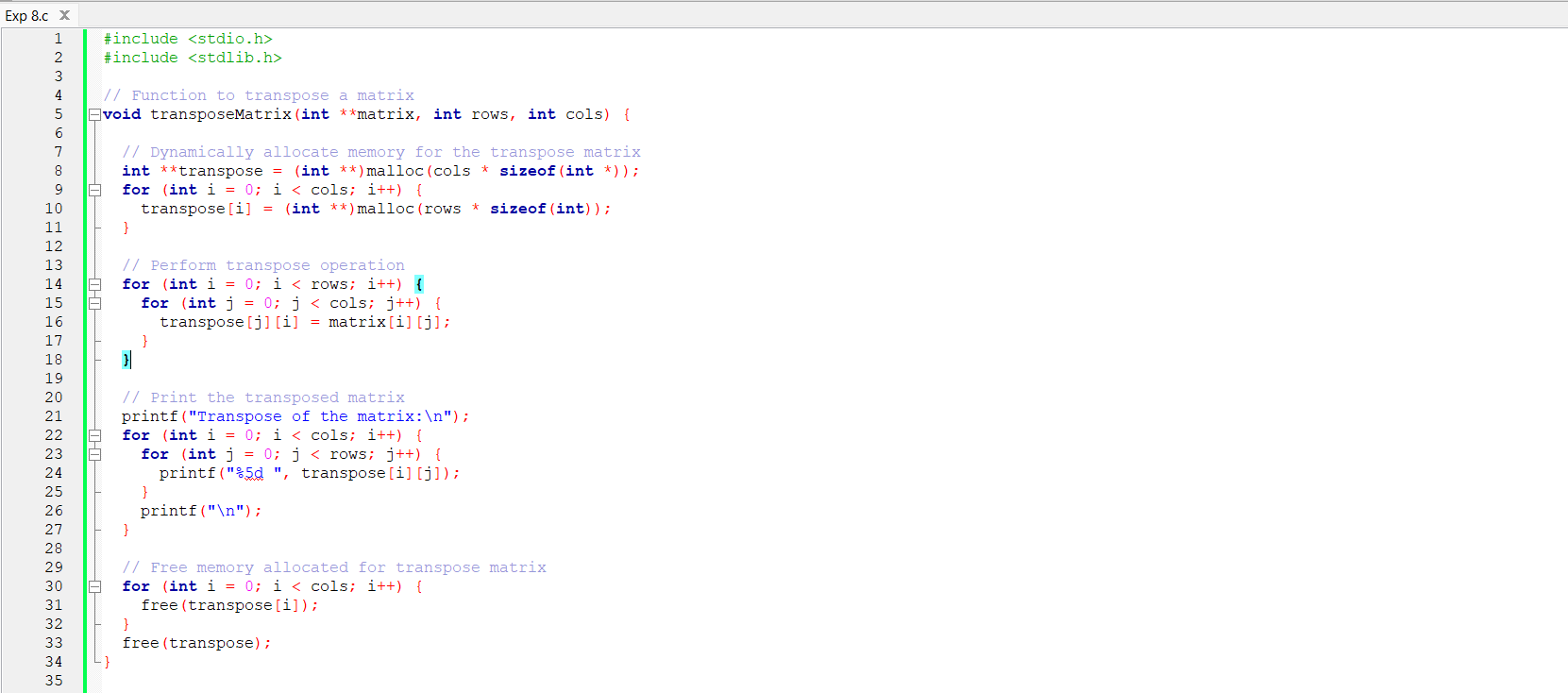
1. Begin.
2. Input the number of rows and columns of the matrix.
3. Dynamically allocate memory for the matrix based on the number of rows and columns.
4. Input matrix elements from the user.
   * For each row:
     + For each column:
       - Prompt the user to enter the element at the current position.
       - Read and store the input element.
5. Display the original matrix.
   * For each row:
     + For each column:
       - Print the element at the current position.
     + Move to the next row and repeat the process.
6. Call the **transposeMatrix** function with the matrix, number of rows, and number of columns as arguments.
7. Within the **transposeMatrix** function:
   * Dynamically allocate memory for the transpose matrix based on the number of columns and rows.
   * Transpose the original matrix by swapping rows and columns.
   * Display the transposed matrix.
   * Free the memory allocated for the transpose matrix.
8. Free the memory allocated for the original matrix.
9. End.

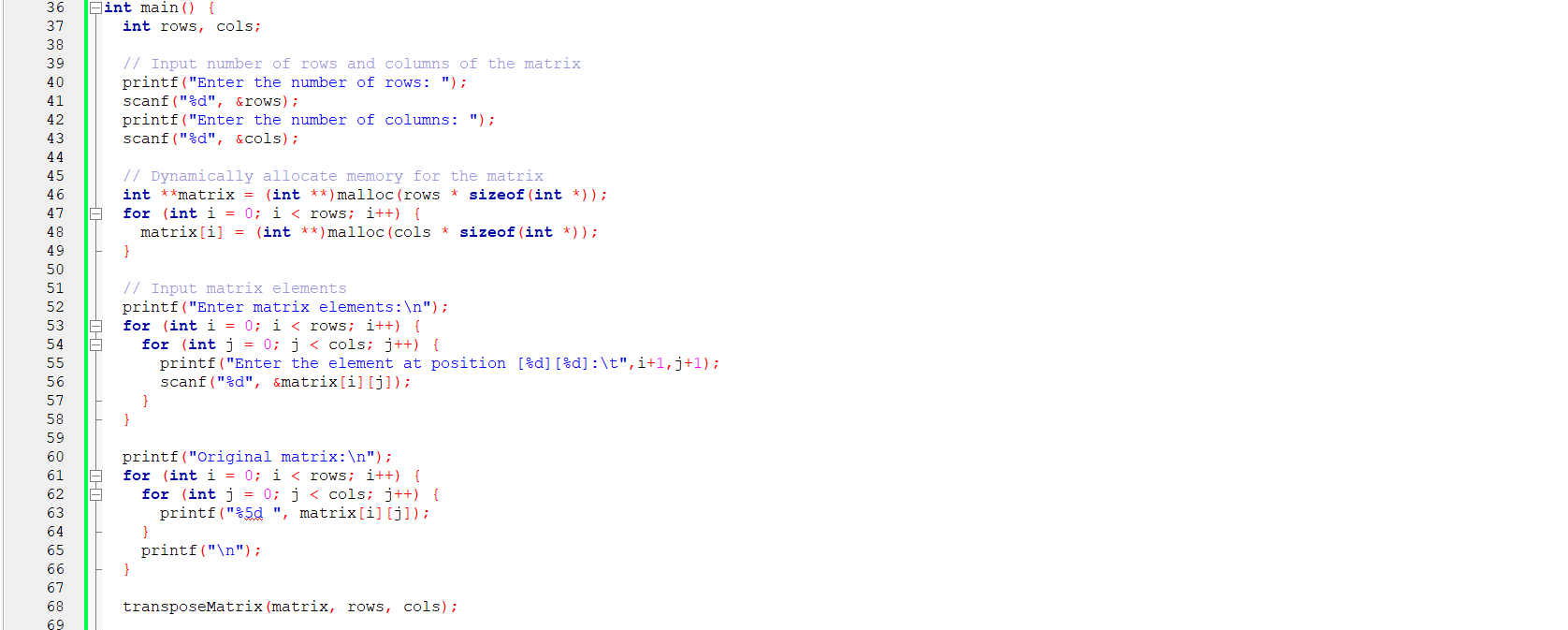
**2) Copying the content of the source file to destination file character by character**

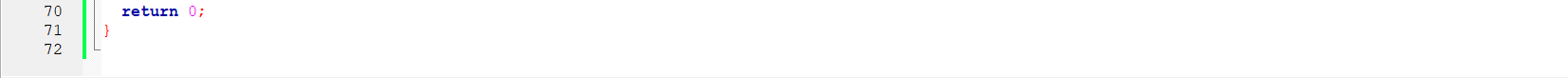
1. Begin.
2. Open the source file in read mode ("source.txt").
   * If the file opening operation fails:
     + Print an error message indicating the failure.
     + Exit the program with a return value of 1.
3. Open the target file in write mode ("destination.txt").
   * If the file opening operation fails:
     + Print an error message indicating the failure.
     + Close the source file.
     + Exit the program with a return value of 1.
4. While there are characters to be read from the source file:
   * Read a character from the source file.
   * Write the character to the target file.
5. Close both the source and target files.
6. Print a success message indicating that the file copy operation was successful.
7. End.

**Implementation details:**

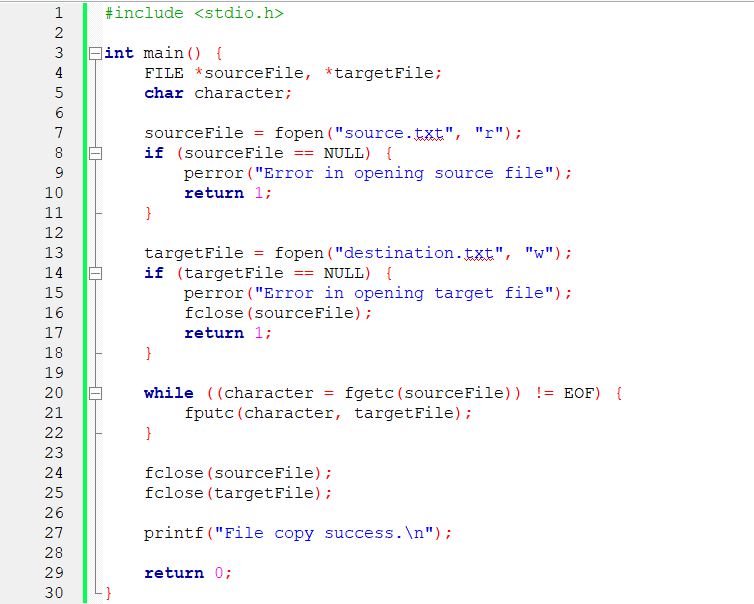
**Q.1) Transpose of a matrix**







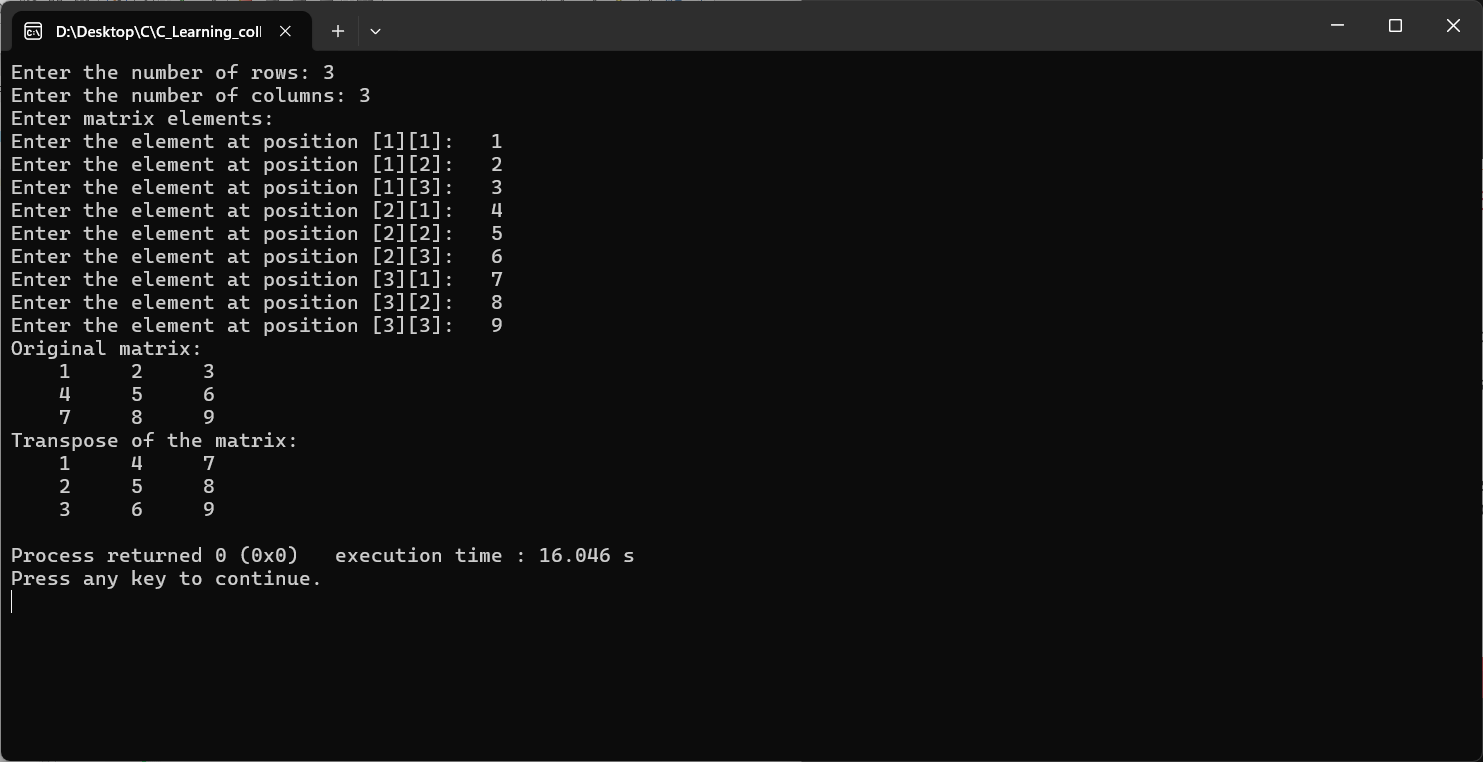
**Q2) Copying the content of source file to destination file character by character**



**Output(s):**

**Q.1)**

**Case 1: When rows and columns are equal**

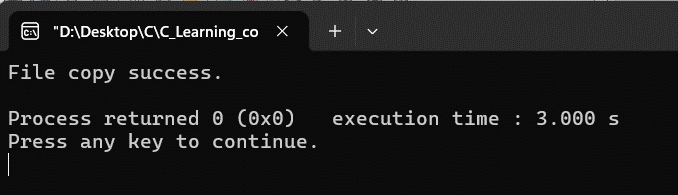


**Case 2: When rows and columns are not equal**

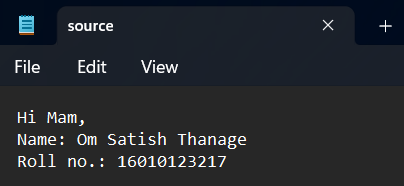


Q.2)

**Terminal output**



**source.txt**



**destination.txt**

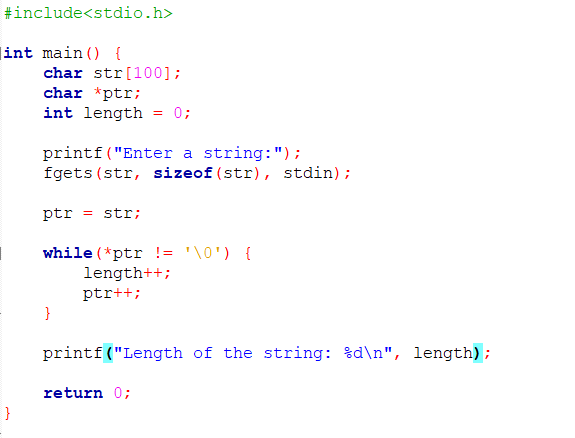
|  |  |
| --- | --- |
| Before | After |
|  |  |

**Conclusion:**

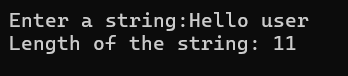
Both programs show basic ideas in C programming, like working with files, managing memory, doing input/output tasks, and handling mistakes. They are organized well and give clear instructions to users. Also, they handle resources like memory and files correctly. In general, these programs are good examples of how to work with files and matrices in C.

**Post Lab Descriptive Questions**

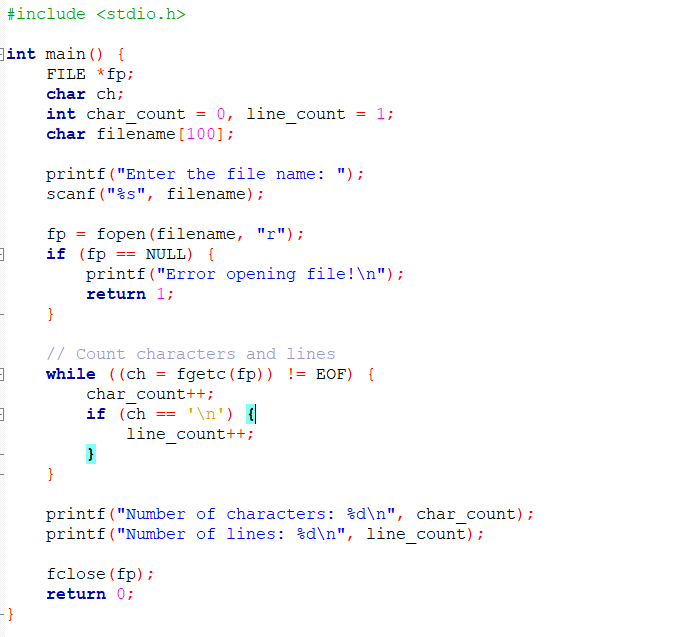
* WAP to accept a string from the user and calculate the length of a given string using pointers.



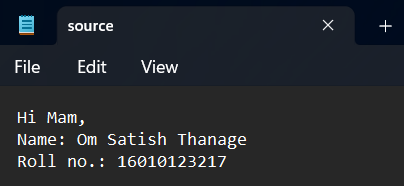
**Output:**



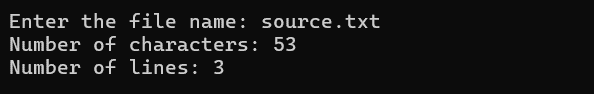
* WAP to count the number of characters and number of lines in a file.



**Text file: Source.txt**



**Output:**



**Date: \_\_\_\_\_\_\_\_\_\_\_\_\_ Signature of faculty in-charge**