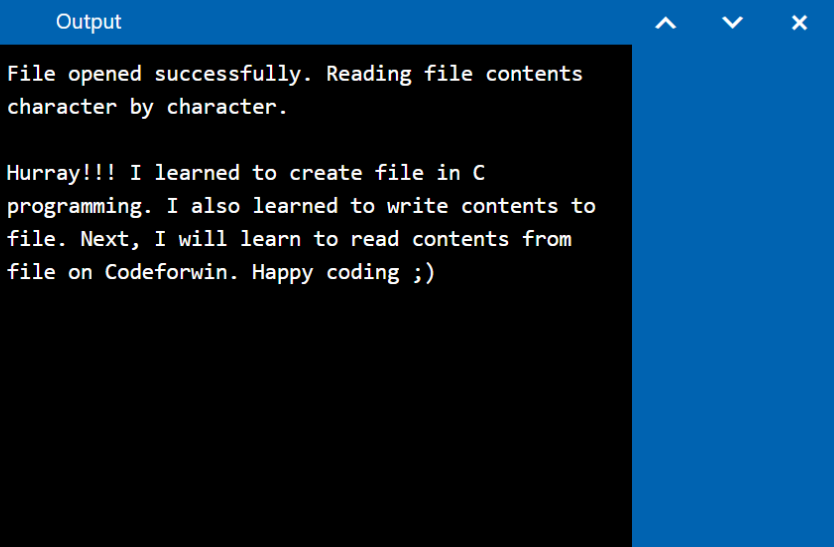
Practical-1

**Using file handling function read the content of file on terminal and write the contents in new file.**

**Source Code:** [**https://github.com/aishiiousness/System-Software/tree/main/File-Handling-Function**](https://github.com/aishiiousness/System-Software/tree/main/File-Handling-Function)

**Output:**

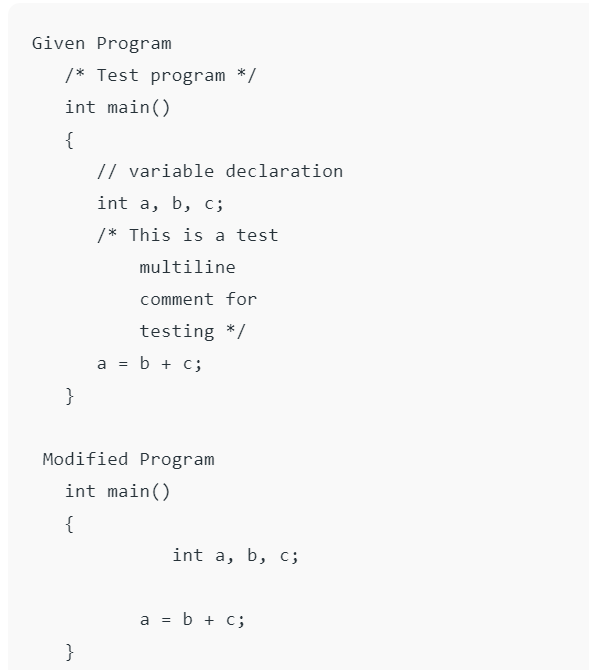


Practical-2

**Aim: Take a file as input and remove comments from the file.**

**Source Code:** [**https://github.com/aishiiousness/System-Software/tree/main/Lexical-Analyser**](https://github.com/aishiiousness/System-Software/tree/main/Lexical-Analyser)

**Output:**

****

Practical-3

**Aim:** Lexical Analyzer: (Two Lab Turn) Write a Program for token specification and recognition. For example, Separation of keywords from identifiers, identify a set of operators etc. Basically, techniques used for lexical analysis are useful in many programming applications other than compilers. Lexical analyzers are pattern recognition engines. Students may work on identifying tokens for a subset of C language constructs.

**Source Code:** [**https://github.com/aishiiousness/System-Software/tree/main/Lexical-Analyser**](https://github.com/aishiiousness/System-Software/tree/main/Lexical-Analyser)

**Output:**

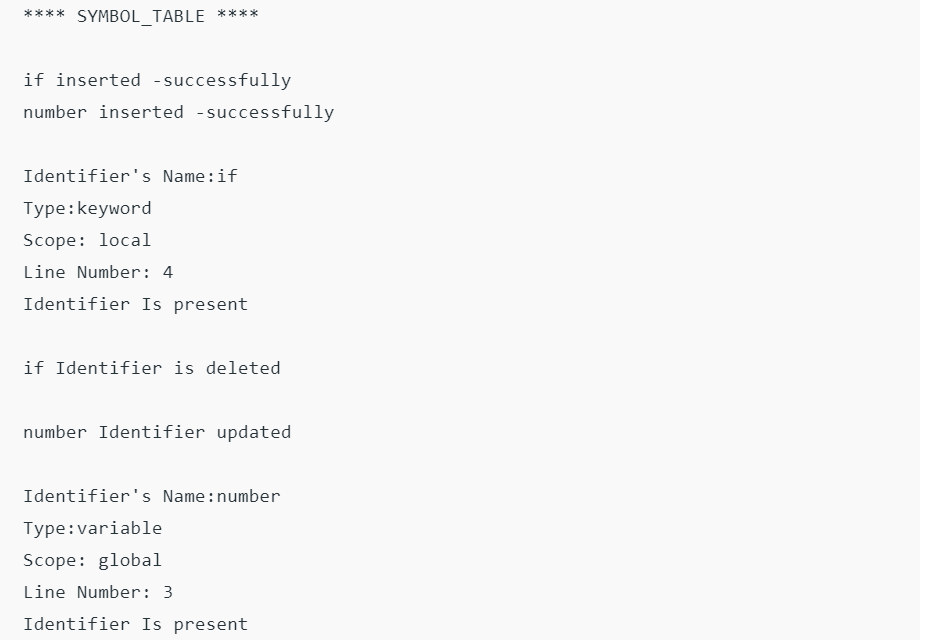
****

Practical-4

**Aim:** Incorporating a symbol table: (One Lab Turn) A data structure called symbol table is used to store information about various source language constructs. During lexical analysis, the character string or lexeme, forming an identifier is saved in a symbol-table entry. Later phases of compiler might add to this entry information such as the type of the identifier, the usage and its position in storage.

**Source Code:** [**https://github.com/aishiiousness/System-Software/tree/main/Symbol-Table**](https://github.com/aishiiousness/System-Software/tree/main/Symbol-Table)

**Output:**

****

Practical-5

**Aim:** Use macro features of C language.

**Source Code:** [**https://github.com/aishiiousness/System-Software/tree/main/Marco-Function**](https://github.com/aishiiousness/System-Software/tree/main/Marco-Function)

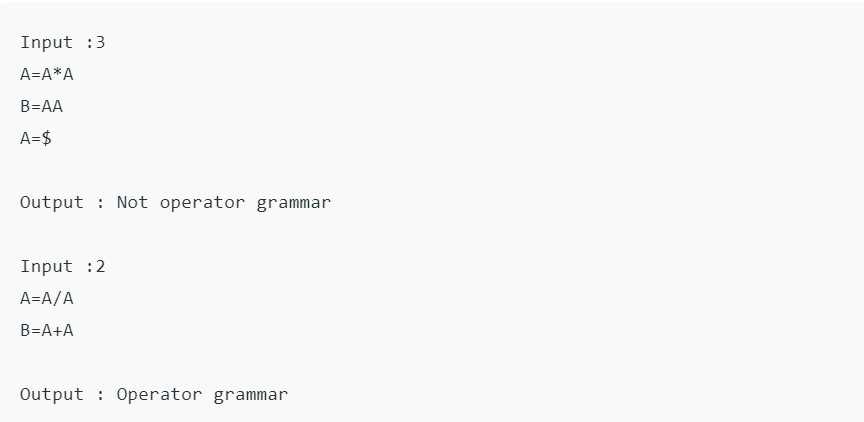
**Output:**

Practical-6

**Aim:** Develop an operator precedence parser for the following grammar. (One lab turn) E-->E+E | E-E | E\*E | id. This grammar generates strings like id-id\*id etc.

**Source Code:** [**https://github.com/aishiiousness/System-Software/tree/main/Operator-Precedence-Parse**](https://github.com/aishiiousness/System-Software/tree/main/Operator-Precedence-Parse)

**Output:**

****

Practical-7

**Aim:** Write a program to generate calculator in Assembly language.

**Source Code:** [**https://github.com/aishiiousness/System-Software/tree/main/Assembly-Language-Calculator**](https://github.com/aishiiousness/System-Software/tree/main/Assembly-Language-Calculator)

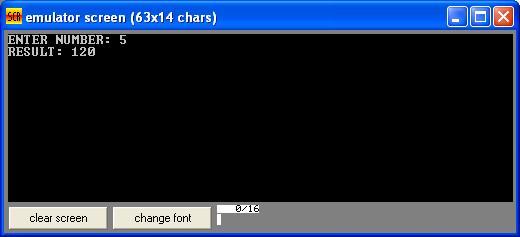
**Output:**

Practical-8

**Aim:** Write a program to find factorial of a given number in Assembly language.

**Source Code:** [**https://github.com/aishiiousness/System-Software/tree/main/Assembly-Language-Factorial**](https://github.com/aishiiousness/System-Software/tree/main/Assembly-Language-Factorial)

**Output:**

****