**PRACTICAL FILE**

**COMPILER DESIGN**

**(CS 604)**

**BE CSE 5TH SEM**

**(GROUP-4)**

****

**University Institute of Engineering and Technology (UIET), Panjab University, Chandigarh, India- 160014**

**Under the guidance of Submitted By**

Dr. Makhan Singh Sir Ojas Arora

Department of Computer Science and Engineering Roll No: UE223073

**INDEX**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SNO.** | **DATE** | **TITLE** | **PAGE NO.** | **REMARKS** |
| 1. | 28-01-2025 | Construction of a DFA with even number of 0s and 1s | 3-4 |  |
| 2. | 04-02-2025 | Write a Program for a lexical analyzer . | 5-6 |  |
| 3. | 18-02-2025 | Given an input string for a regular expression. Obtain the corresponding NFA with null moves. | 7-8 |  |
| 4. | 18-03-2025 | To implement various LEX Programs. | 9-10 |  |
| 5. | 25-03-2025 | To implement a Recursive Descent Parser. | 11-12 |  |
| 6. | 01-04-2025 | To implement the First Follow and LL1 Parser. | 13-14 |  |
| 7. | 08-04-2025 | Design a Program to implement Bottom-Up Parser (Operator Precedence Parser). | 15-16 |  |
| 8. | 08-04-2025 | To implement LR Parser. | 17-19 |  |
| 9. | 08-04-2025 | To implement Syntax-directed Translation. | 20-21 |  |

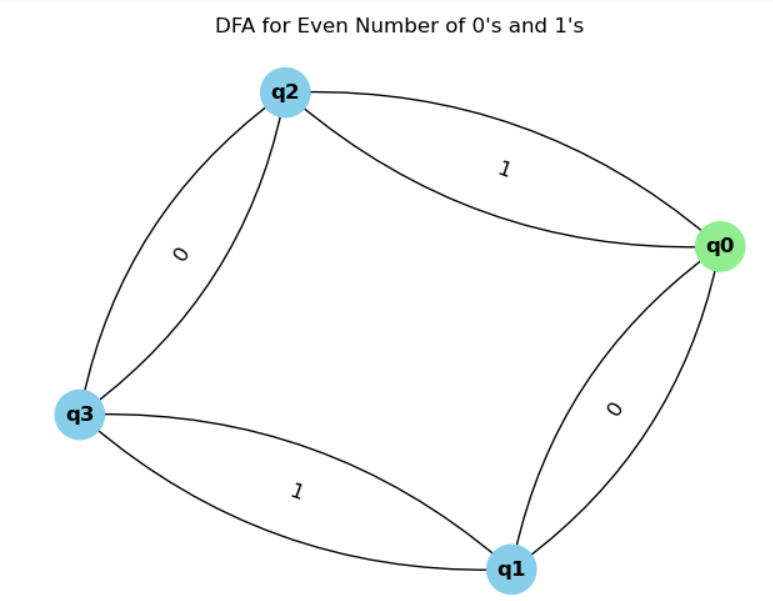
**Practical 1**

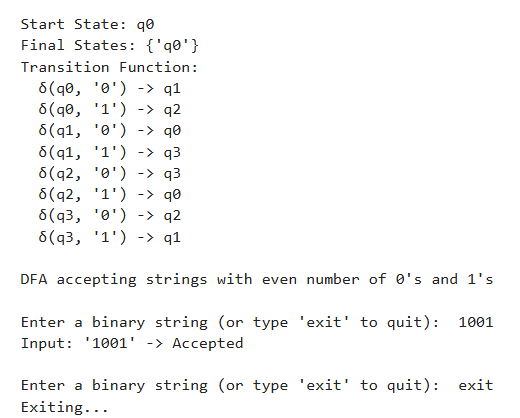
**Aim**

Construct a DFA with even number of 0s and 1s

**Code for Construction of a DFA with even number of 0s and 1s**



**Output**



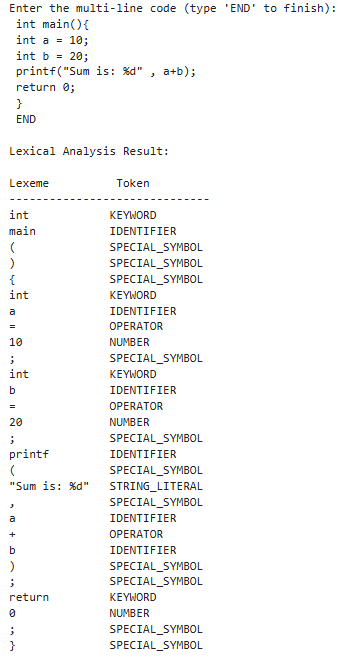
**Practical 2**

**Aim**

Program for a Lexical Analyzer

**Code for Implementation of Program for a Lexical Analyzer**

**Output**



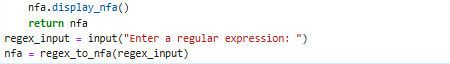
**Practical 3**

**Aim**

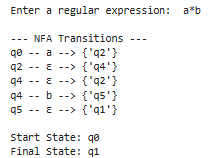
To convert input string for a regular expression to obtain the corresponding NFA with null moves.

**Code for Implementation of RE input string to NFA with null moves**





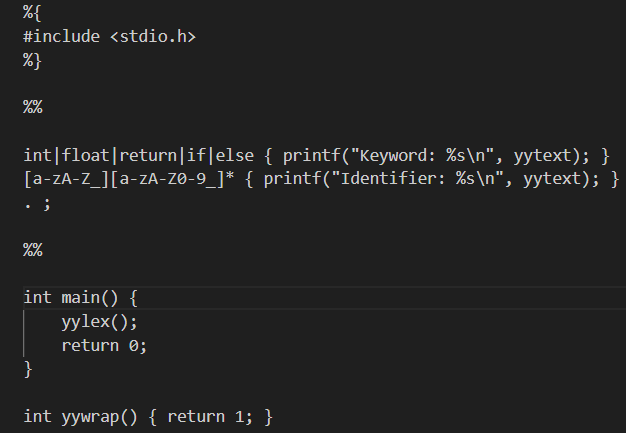
**Output**



**Practical 4**

**Aim**

To implement various LEX Programs.

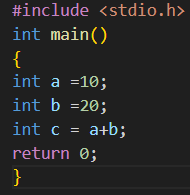
**Code for Implementation of various LEX Programs**

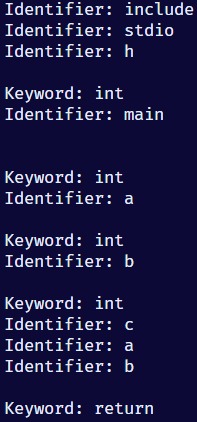
lex.l File

Install the Flex Tool in your PC with its Path in the environment variables and then run the command flex lex.l in the terminal . This will lead to the Generation of File lex.yy.c

After the generation of lex.yy.c use the command gcc ./lex.yy.c -o a.exe to use the C Compiler to convert lex.yy.c to a.exe or a.out .

After that in order to generate Tokens from the a.exe File run the command Get-Content ./Program.c | ./a.exe



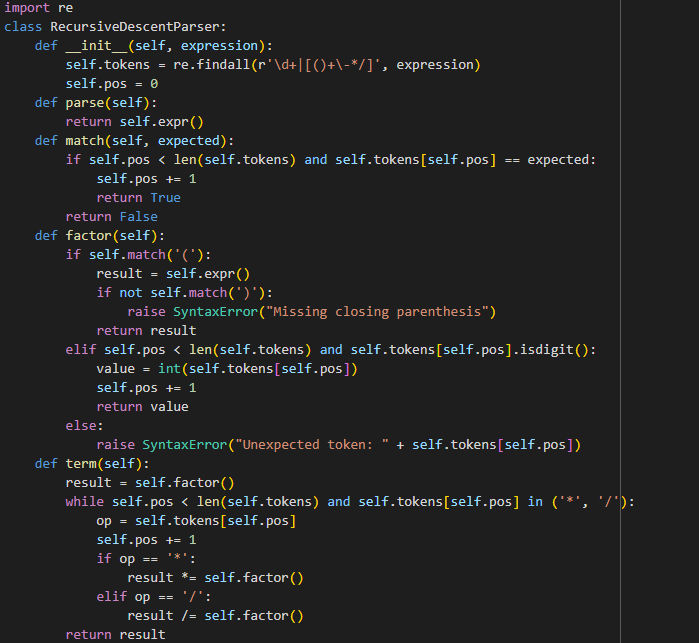
Program.c

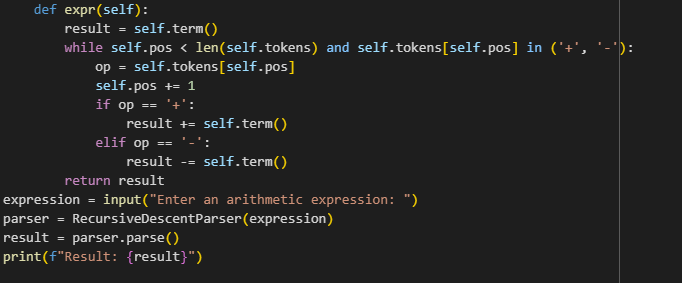
Tokens Generated from a.exe

**Practical 5**

**Aim**

To implement a Recursive Descent Parser.

**Code for Implementation of Recursive Descent Parser**



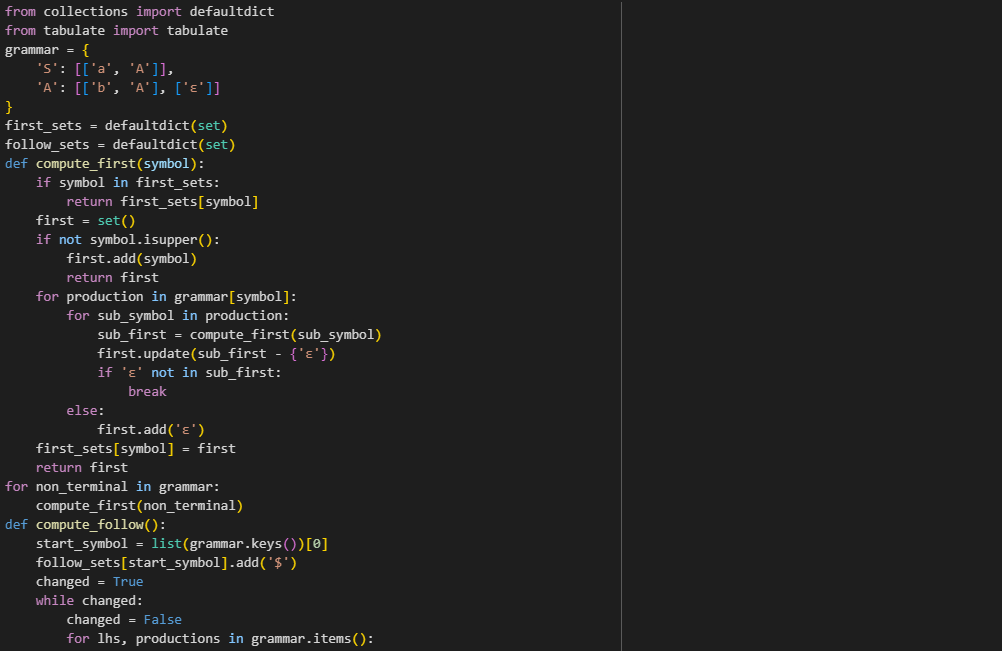
**Output**

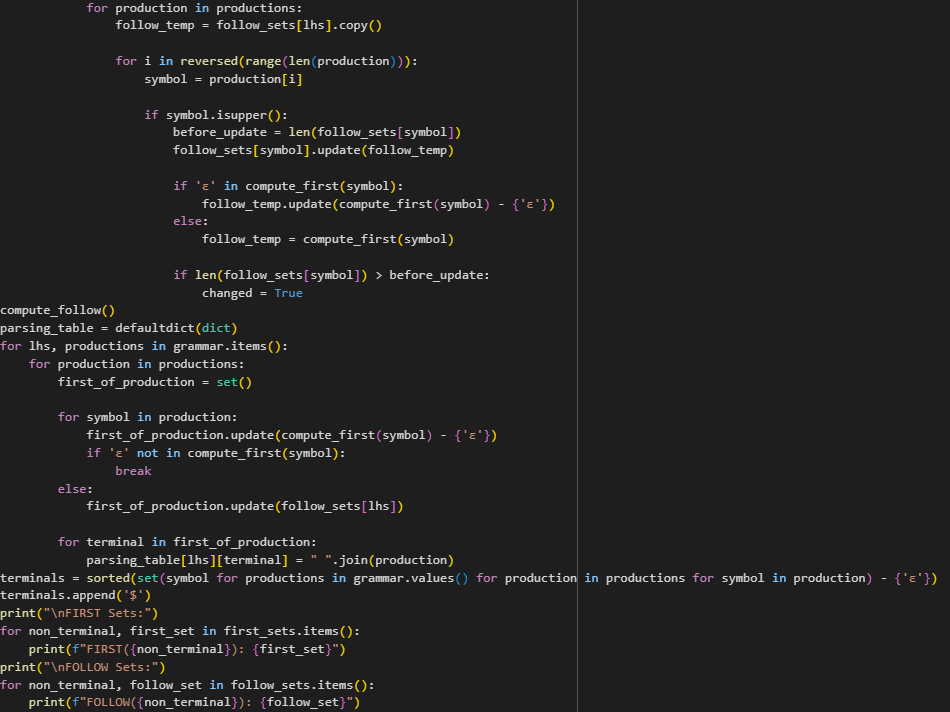


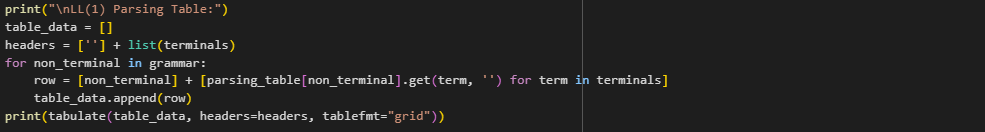
**Practical 6**

**Aim**

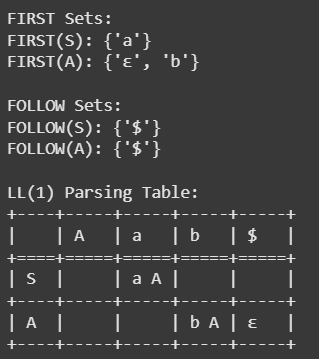
To implement the First Follow and LL1 Parser.

**Code for Implementation of the First Follow and LL1 Parser**





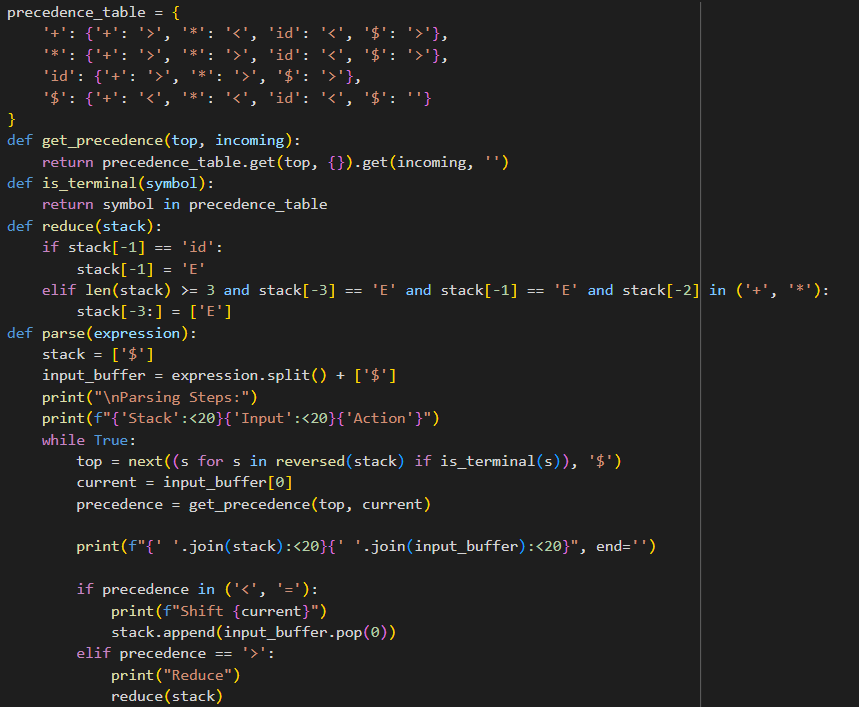
**Output**

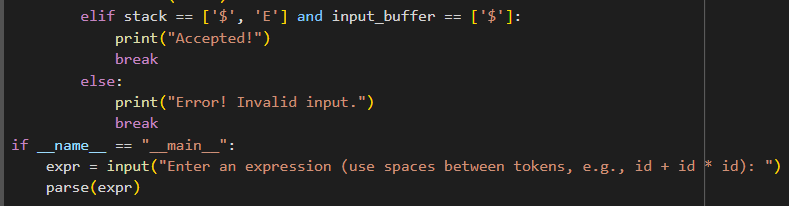


**Practical 7**

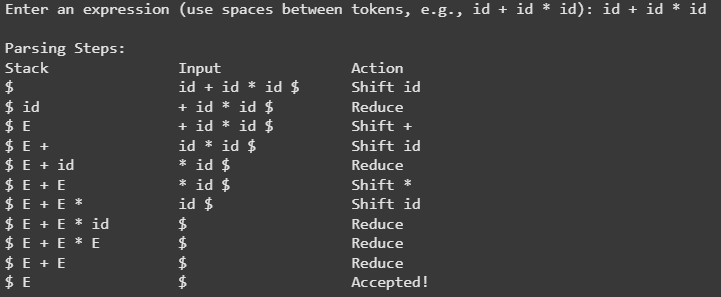
**Aim**

Design a Program to implement Bottom-Up Parser (Operator Precedence Parser)

**Code for Implementation of Bottom-Up Parser (Operator Precedence Parser)**



**Output**



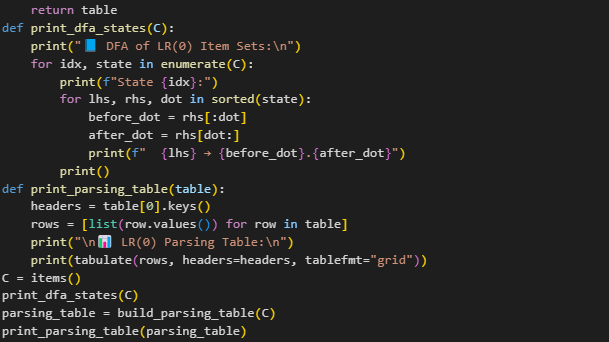
**Practical 8**

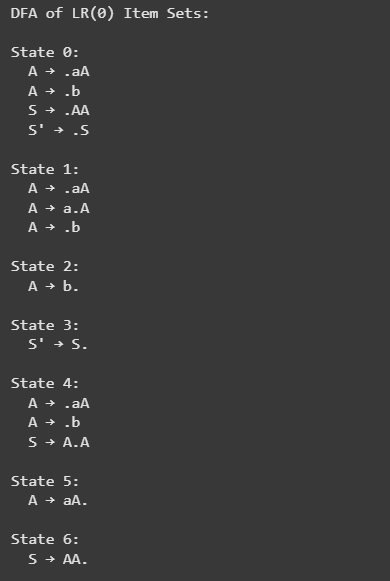
**Aim**

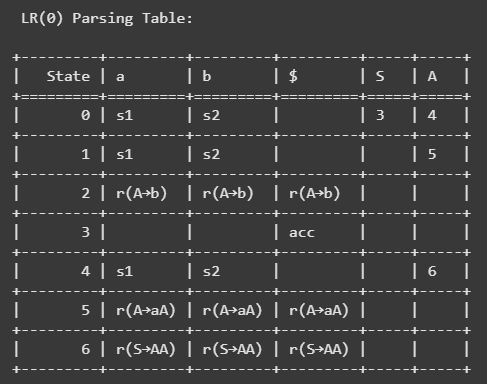
To implement LR Parser

**Code for Implementation of LR Parser**





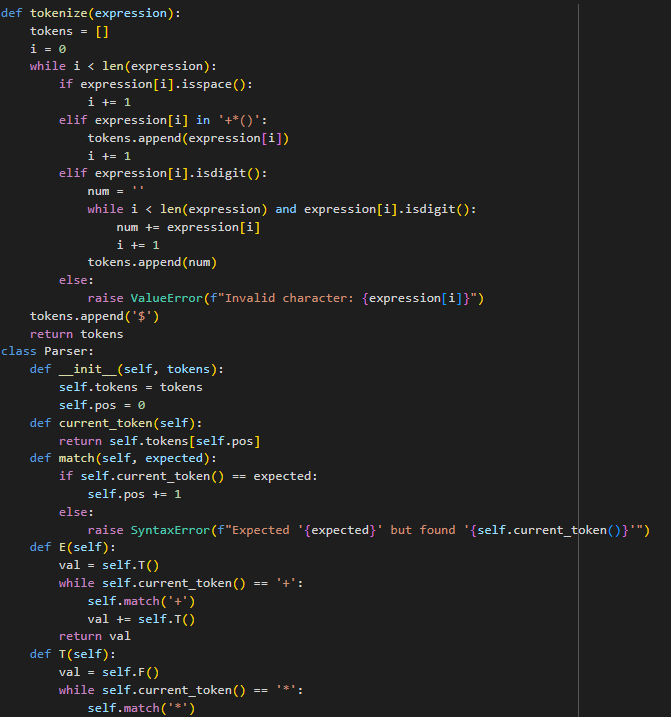
**Output**

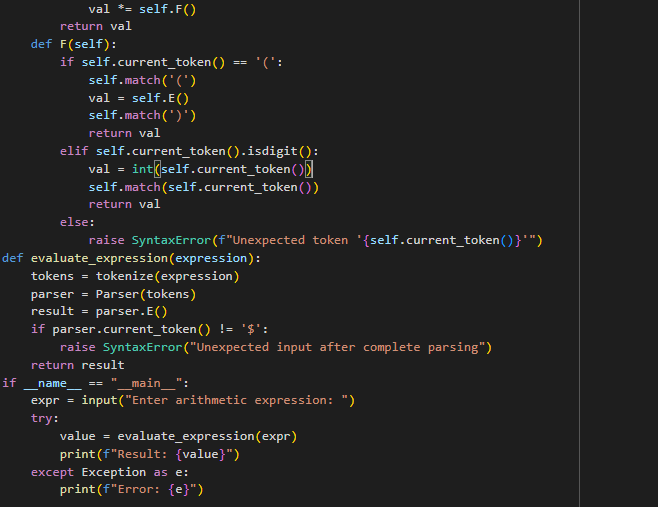
****

**Practical 9**

**Aim**

Implementation of Syntax-Directed Translation.

**Code for Implementation of Syntax-Directed Translation**



**Output**

