

**Faculty of Technology and Engineering**

**U & P U. Patel Department of Computer Engineering**

**Practical Performa**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Academic Year | : | 2024-25 | Semester | : | 6 |
| Course code | : | CE365 | Course name | : | Compiler Construction Design of Language Processor |

**Practical 1**

**Aim: String Validation Against Regular Expression**

**Objective:** To implement a program that validates a user-input string against the regular expression a\*bb. The program should determine whether the input string is valid or invalid based on the defined pattern.

**Screen Shot of Output:**

**A black background with white text

AI-generated content may be incorrect.**

**A close-up of a black screen

AI-generated content may be incorrect.**

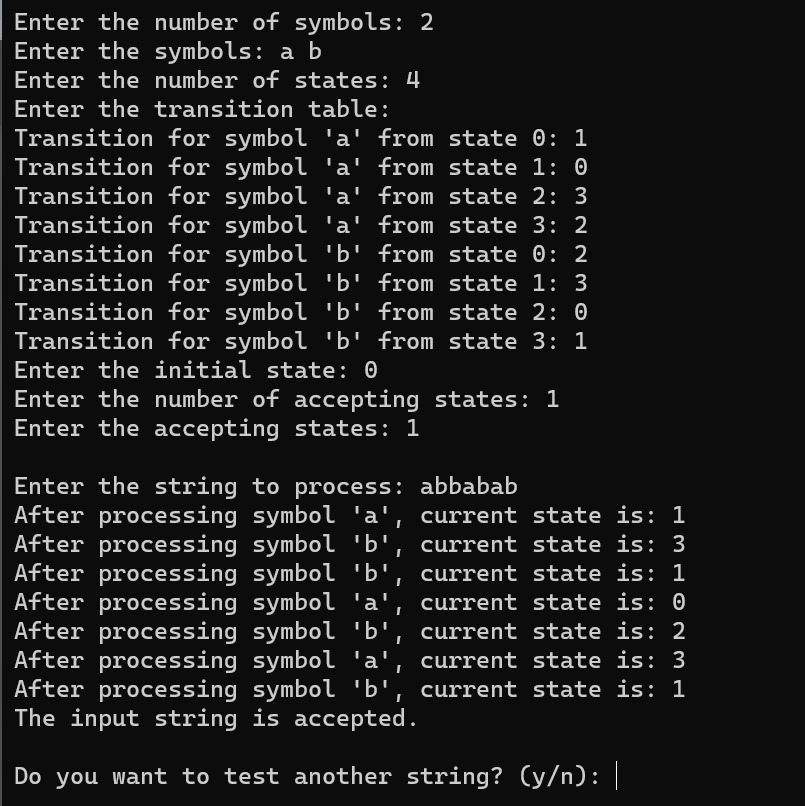
**Conclusion:**

**Practical 2**

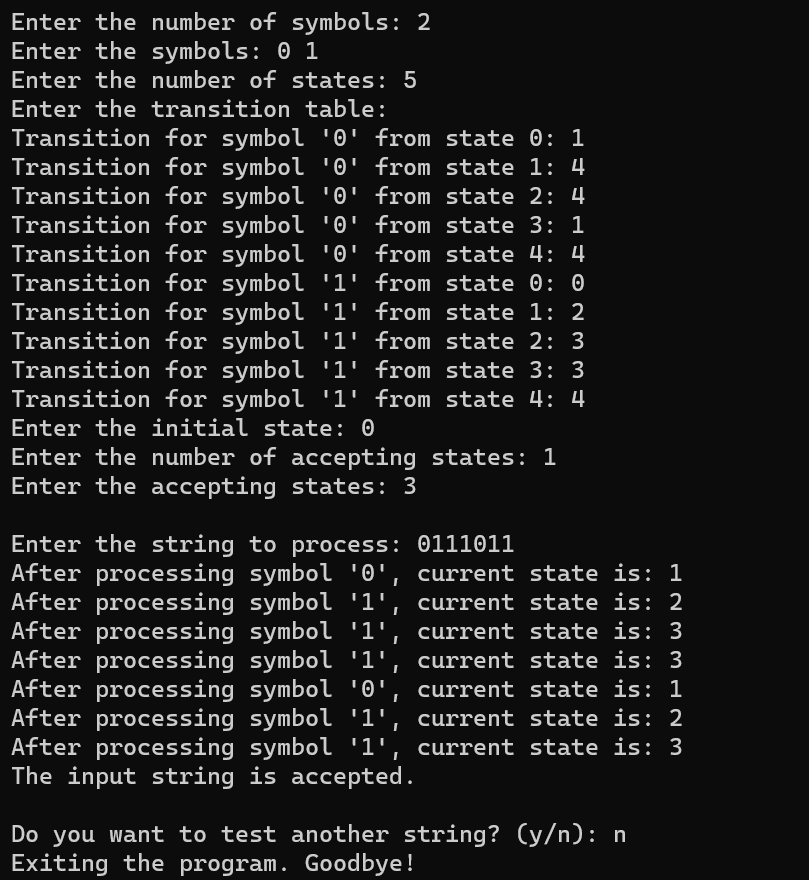
**Aim: String Validation Using Finite Automata**

**Objective:** To implement a program that validates a given string against rules defined in terms of finite automata.

**Screen Shot of Output:**



**String over 0 and 1 where every 0 immediately followed by 11**

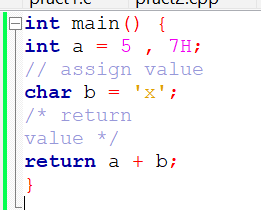
****

**Conclusion:**

**Practical 3**

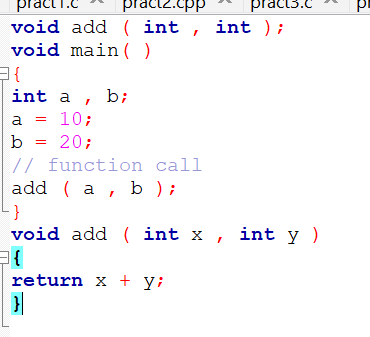
**Aim: Implementation of a Lexical Analyzer for C Language Compiler**

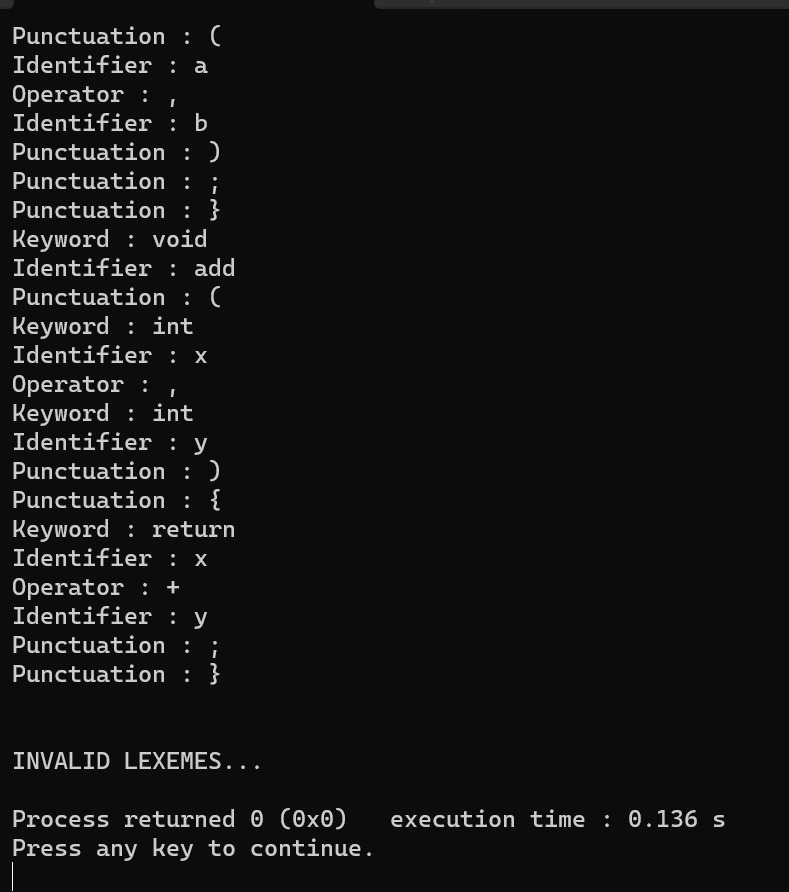
**Objective:** To design and implement a lexical analyser, the first phase of a compiler, for the C programming language. The lexical analyser should perform the following tasks: (1) tokenizing the input string (2) removing comments (3) removing white spaces (4) entering identifiers into the symbol table (5) generating lexical errors.

****

**Screen Shot of Output:**



****

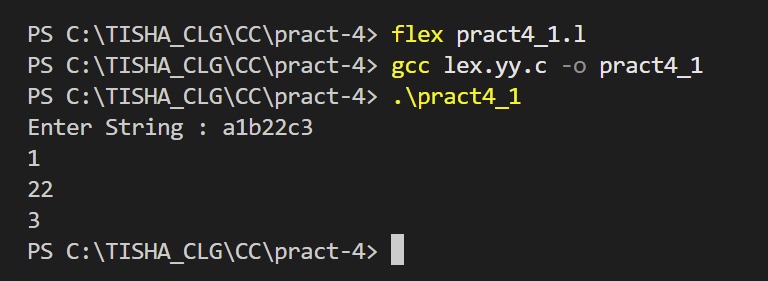


**Conclusion:**

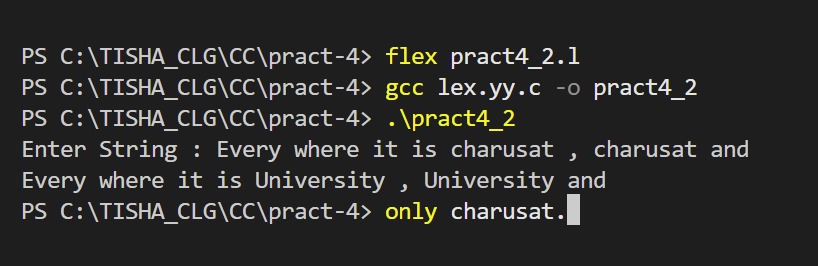
**Practical 4**

**Aim: String validation using Lax tool**

**Objective - 1** Write a program to identify and extract all numbers from input string and display them one by one in new line.



**Objective - 2** Write a program to replace the word "charusat" with “university” in the input text.

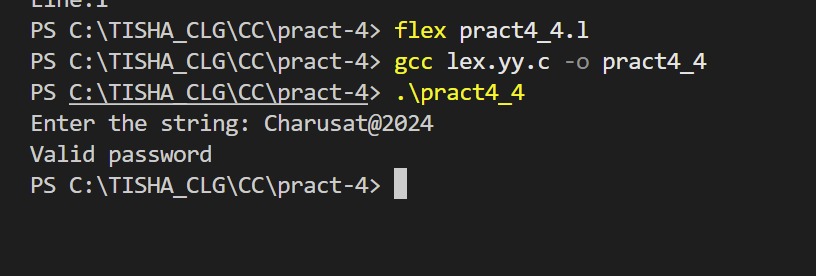


**Objective – 3** Write a program to count number of characters, word and lines from the input file.

A screenshot of a computer program

AI-generated content may be incorrect.

**Objective – 4** Write a program which validate the password as per given rules. ➢ length can be 9 to 15 characters ➢ includes lower case letter, upper case letter, digit, symbols (\*, ; # $ @) ➢ minimum count for each category must be o



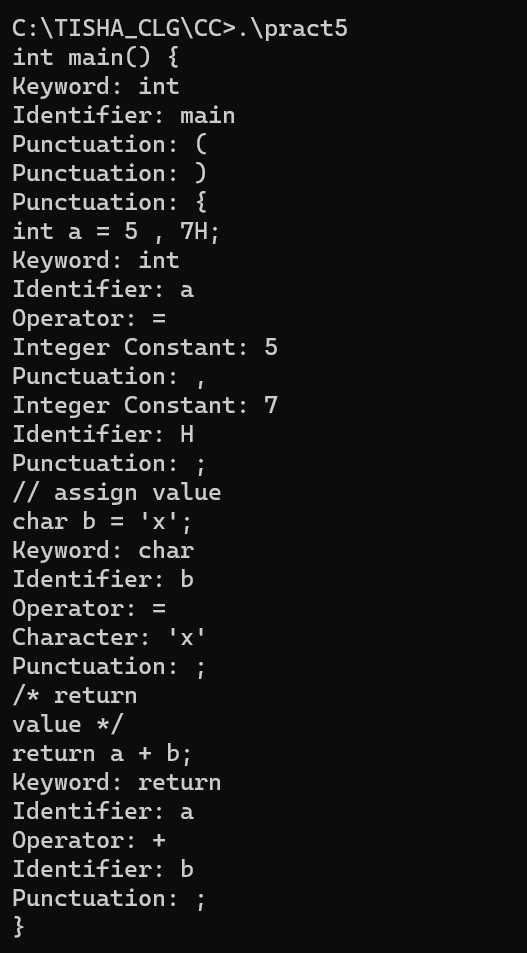
**Conclusion:**

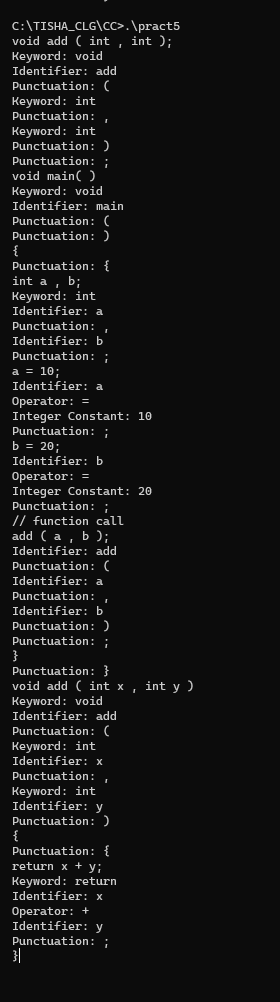
**Practical 5**

**Aim: Implementation of a Lexical Analyzer for C Language Compiler**

**Objective**: To design and implement a lexical analyser to perform 1st, 2nd, 3rd, and 5th task as per the list given in practical 2.

**Screen Shot of Output:**

****

****

**Conclusion:**

**Practical 6**

**Aim: String validation using Recursive Descent Parsing (RDP)**

**Objective**: Implement a Recursive Descent Parser (RDP) to validate an input string against the given grammar. S → ( L ) | a L → S L’ L’ → , S L’ | ϵ

**Screen Shot of Output:**

A computer screen with text

AI-generated content may be incorrect.

**Conclusion:**

**Practical 7**

**Aim: Computing First and Follow Sets for a Context-Free Grammar (CFG)**

**Objective**: Develop a program computes the First and Follow sets for all non-terminal symbols in for the below given grammar.

S → A B C | D

A → a | ε B → b | ε

C → ( S ) | c

D → A C

**Screen Shot of Output:**

A computer screen with a black background

AI-generated content may be incorrect.

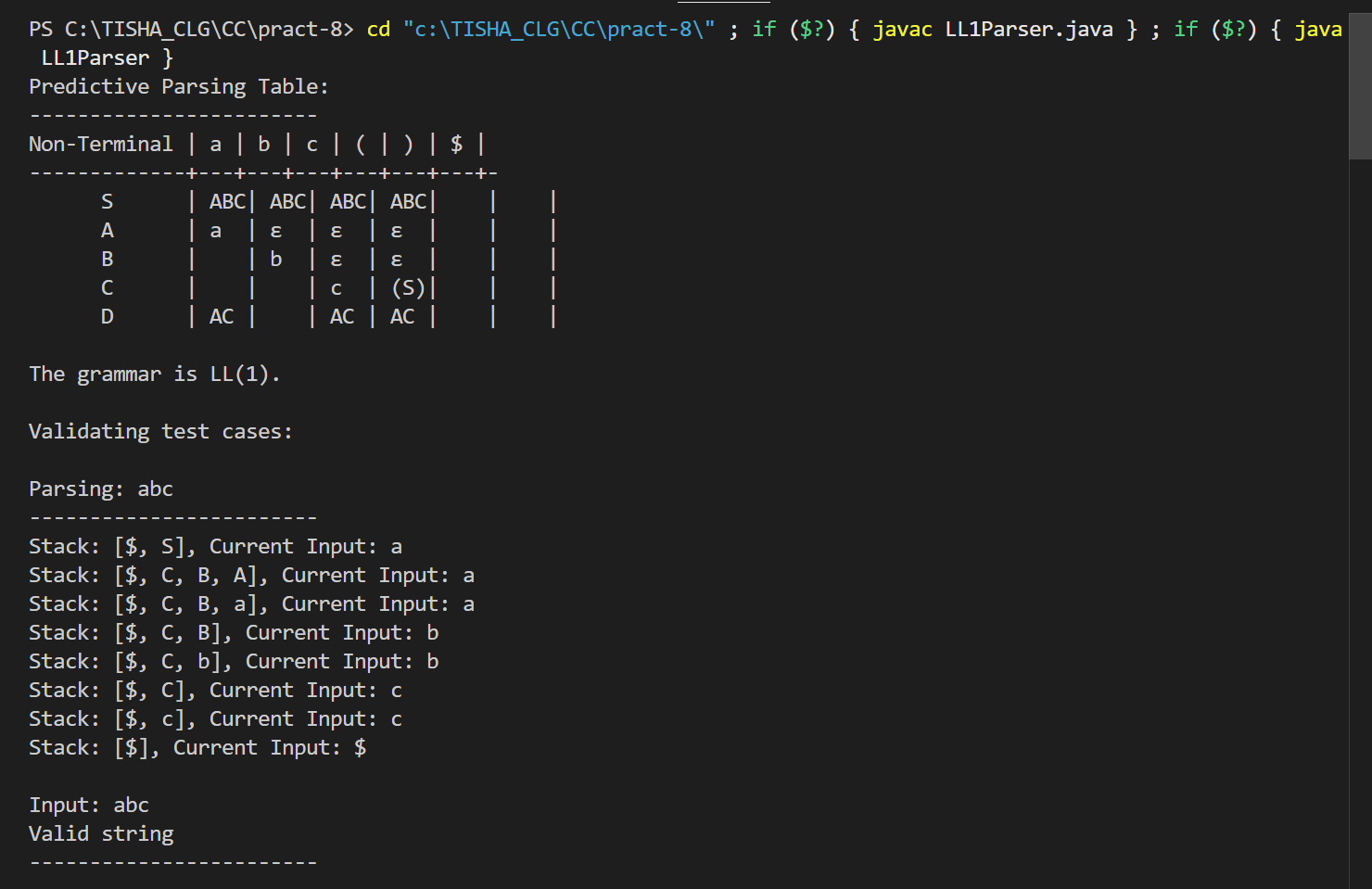
**Conclusion:**

**Practical 8**

**Aim: Predictive Parsing Table Construction and LL(1) Grammar Validation**

**Objective**: Develop a program to construct a predictive parsing table for the given grammar. The program should analyse the generated parsing table to determine whether the grammar is LL(1) or not. If the grammar is LL(1), the program should also validate an input string against the given grammar.

**Screen Shot of Output:**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Conclusion:**

**Practical 9**

**Aim: String parsing using YACC**

**Objective**: Develop a YACC program to validate input strings based on the given grammar. The program should parse the string using the grammar rules and determine whether the string is valid or invalid.

S → i E t S S' | a

S' → e S | ε

E → b

**Screen Shot of Output:**

A computer screen shot of a computer code

AI-generated content may be incorrect.

**Conclusion:**

**Practical 10**

**Aim: Evaluating Arithmetic Expression with Bottom-Up Approach Using SDD**

**Objective**: Develop a program to evaluate arithmetic expressions containing operators using a bottom-up parsing approach and below given Syntax-Directed Definitions (SDD) for semantic evaluation. The program will compute the result of the expression by building a parse tree using and will incorporate semantic rules to evaluate sub-expressions during parsing A table of math equations

AI-generated content may be incorrect.

**Screen Shot of Output:**

A screen shot of a computer program

AI-generated content may be incorrect.

**Conclusion:**

**Practical 11**

**Aim: Generate Intermediate Code Using Quadruple Table**

**Objective**: Develop a program that break down the input string according to the grammar and produce a sequence of quadruples representing the intermediate code for the expression.

E → E + T | E – T | T

T → T \* F | T / F | F

F → (E) | digit

**Screen Shot of Output:**

A computer screen shot of a black screen

AI-generated content may be incorrect.

**Conclusion:**

**Practical 12**

**Aim: Code Optimization Using Constant Folding**

**Objective**: Develop a program that identifies constant expressions at compile-time and replaces them with their evaluated results to enhance execution efficiency.

**Screen Shot of Output:**

A computer screen with text and numbers

AI-generated content may be incorrect.

**Conclusion:**