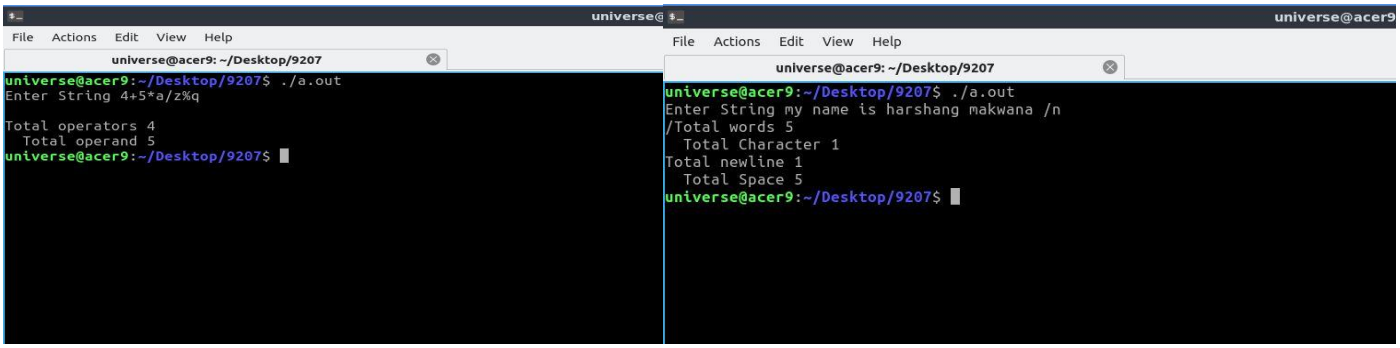
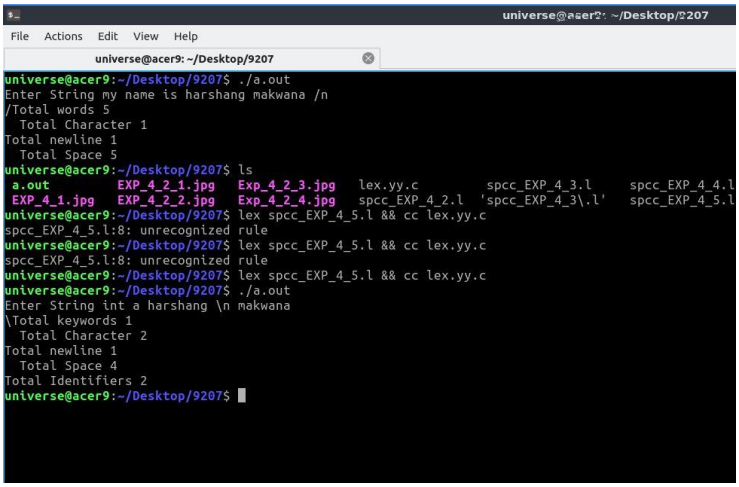
FR. CONCEICAO RODRIGUES COLLEGE OF ENGINEERING

System Programming and Compiler Construction

VI Semester (Computer) **Academic Year: 22-23**





Aim : Study of Parser generator tool – Yacc

Leraning Objective:

Theory:

Parser for a grammar is a program which takes in the language string as its input and produces either a corresponding parse tree or a error. Syntax of a Language The rules which tells whether a string is a valid program or not are called the syntax Semantic s of Language The rules which give meaning to programs are called the semantic of a language Tokens When a string representing a program is broken into sequence of substrings, such that each substring represents a constant, identifier, operator, keyword etc of the language, these substrings are called the tokens of the language.

Lexical Analysis

1. % union It defines the Stack type for the Parser.

   It is union of various datas/structures/objects.

2. % token These are the terminals returned by the yylex function to the yacc. A token cal also have type associated with it for good type checking and syntax directed translation. A type of a token can be specified as % token <stack member> tokenName.

3. %type The type of non-terminal symbol in the grammar rule can be specified with this. The format is %type <stack member> non termainal.

4. % noassoc Specifies that there is no associativity of a terminal symbol.

5. % left Specifies the left associativity of a terminal symbol.

6. % right Specifies the right associativity of a terminal symbol.

7. % start specifies the L.H.S. non-terminal symbol of a production rule which specifies starting point of grammar rules.

8. % prac changes the precedence level associated with a particular rule to that of the following token name or literal.

The Grammar rules are specified as follows:

Context free grammar production-

p-> AbC

Yacc Rule-

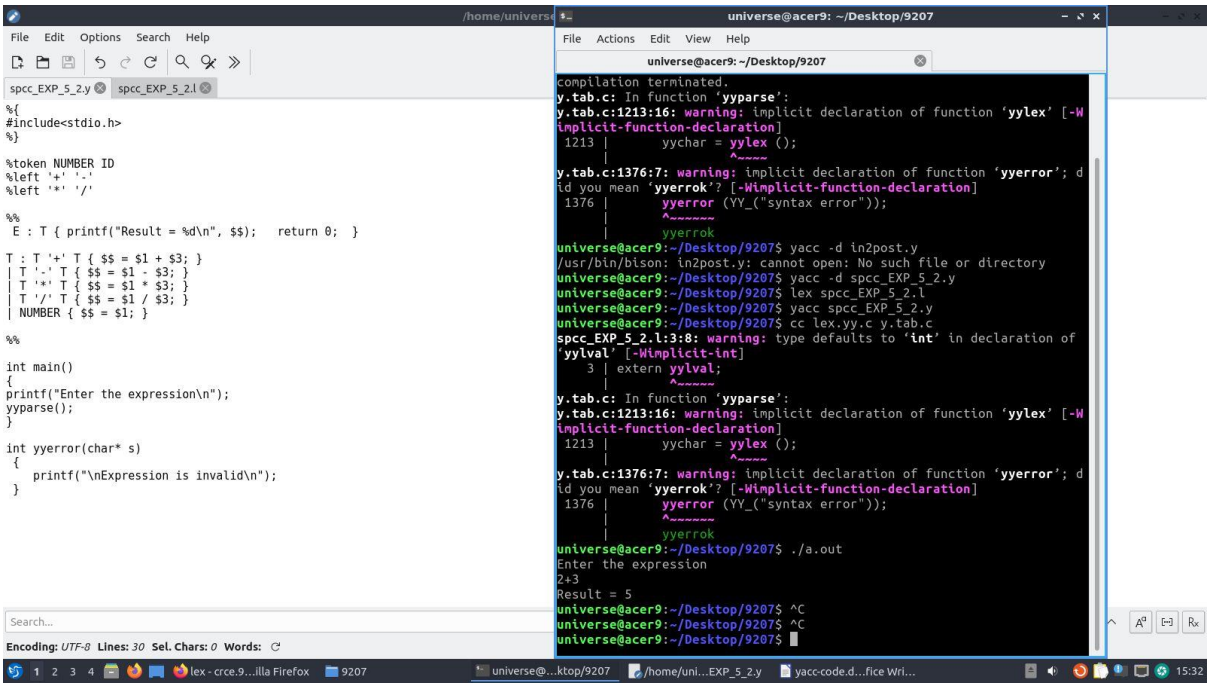
P: A b C { /* ‘C’ actions*/}

lex.yy.o The object file for the **lex.yy.c** source file

a.out The executable program file

1. To then run the program directly from the **a.out** file, enter:
2. `$ a.out`

Output :



Postlab:

1. **Write the structure of Lex**
2. **Write the structure of Yacc**