

FR. CONCEICAO RODRIGUES COLLEGE OF ENGINEERING

System Programming and Compiler Construction

VI Semester (Computer)

Academic Year: 22-23

```
universe@acer9: ~/Desktop/9207
universe@acer9:~/Desktop/9207$ ./a.out
Enter String 4+5*a/z%q

Total operators 4
Total operand 5
universe@acer9:~/Desktop/9207$

universe@acer9: ~/Desktop/9207
universe@acer9:~/Desktop/9207$ ./a.out
Enter String my name is harshang makwana /n
//Total words 5
Total Character 1
Total newline 1
Total Space 5
universe@acer9:~/Desktop/9207$

universe@acer9: ~/Desktop/9207
universe@acer9:~/Desktop/9207$ ./a.out
Enter String my name is harshang makwana /n
//Total words 5
Total Character 1
Total newline 1
Total Space 5
universe@acer9:~/Desktop/9207$ ls
a.out EXP_4_2_1.jpg Exp_4_2_3.jpg lex.yy.c spcc_EXP_4_3.l spcc_EXP_4_4.l
EXP_4_1.jpg EXP_4_2_2.jpg Exp_4_2_4.jpg spcc_EXP_4_2.l 'spcc_EXP_4_3.l' spcc_EXP_4_5.l
universe@acer9:~/Desktop/9207$ lex spcc_EXP_4_5.l && cc lex.yy.c
spcc_EXP_4_5.l:8: unrecognized rule
universe@acer9:~/Desktop/9207$ lex spcc_EXP_4_5.l && cc lex.yy.c
spcc_EXP_4_5.l:8: unrecognized rule
universe@acer9:~/Desktop/9207$ lex spcc_EXP_4_5.l && cc lex.yy.c
universe@acer9:~/Desktop/9207$ ./a.out
Enter String int a harshang \n makwana
\\Total keywords 1
Total Character 2
Total newline 1
Total Space 4
Total Identifiers 2
universe@acer9:~/Desktop/9207$
```

Aim : Study of Parser generator tool – Yacc

Lernaning 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 can 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 terminal.
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. % prec 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 :

```

/home/universe@acer9: ~/Desktop/9207
File Edit Options Search Help
spcc_EXP_5_2.y spcc_EXP_5_2.l
%{
#include <stdio.h>
%}

%token NUMBER ID
%left '+' '-'
%left '*' '/'

%%
E : T { printf("Result = %d\n", $$); return 0; }

T : T '+' T { $$ = $1 + $3; }
  T '-' T { $$ = $1 - $3; }
  T '*' T { $$ = $1 * $3; }
  T '/' T { $$ = $1 / $3; }
  NUMBER { $$ = $1; }

%%

int main()
{
    printf("Enter the expression\n");
    yyparse();
}

int yyerror(char* s)
{
    printf("\nExpression is invalid\n");
}

compilation terminated.
y.tab.c: In function 'yyparse':
y.tab.c:1213:16: warning: implicit declaration of function 'yylex' [-Wimplicit-function-declaration]
1213 | yychar = yylex ();
      |           ^~~~~
y.tab.c:1376:7: warning: implicit declaration of function 'yyerror'; did you mean 'yyerror'? [-Wimplicit-function-declaration]
1376 | yyerror (YY_("syntax error"));
      | ^~~~~~
      | yyerror
universe@acer9:~/Desktop/9207$ yacc -d in2post.y
/usr/bin/bison: in2post.y: cannot open: No such file or directory
universe@acer9:~/Desktop/9207$ yacc -d spcc_EXP_5_2.y
universe@acer9:~/Desktop/9207$ lex spcc_EXP_5_2.l
universe@acer9:~/Desktop/9207$ yacc spcc_EXP_5_2.y
universe@acer9:~/Desktop/9207$ cc lex.yy.c y.tab.c
spcc_EXP_5_2.l:3:8: warning: type defaults to 'int' in declaration of 'yyval' [-Wimplicit-int]
3 | extern yyval;
  |           ^~~~~
y.tab.c: In function 'yyparse':
y.tab.c:1213:16: warning: implicit declaration of function 'yylex' [-Wimplicit-function-declaration]
1213 | yychar = yylex ();
      |           ^~~~~
y.tab.c:1376:7: warning: implicit declaration of function 'yyerror'; did you mean 'yyerror'? [-Wimplicit-function-declaration]
1376 | yyerror (YY_("syntax error"));
      | ^~~~~~
      | yyerror
universe@acer9:~/Desktop/9207$ ./a.out
Enter the expression
2+3
Result = 5
universe@acer9:~/Desktop/9207$ ^C
universe@acer9:~/Desktop/9207$ ^C
universe@acer9:~/Desktop/9207$
  
```

Postlab:

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