Dr. Vishwanath Karad MIT WORLD PEACE UNIVERSITY | PUNE
TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

School of CET

TY BTech CSE

Lab Assignment - System Software and Compiler Lab(CS331) LCA Component

Alok Bhawanakr
PA06
Panel 1
Batch A

Question 1

Structure of LEX specificaiton :

declaration
%%
pattern specification
%%
support routines

Declaration: This section contains C code and majorly is made of variable declaration and library imports. This might also contain some macros for regular expression.

Pattern Specification: In this part, we define the regular expression to identify the tokens with C code perform the apporpriate processes when the token in recognized.
Eg : regular expression {C code ;}

Support Routine: In this section, we write some C codes which include some support and the main function that calls the lex functions and executes the program.

Structure of YACC specification :

declaration
%%
grammar rules
%%
subroutines

Declaration: Serves the same purpose as the declaration segment in the lex file. This also include the declaration of tokens form the LEX file.

Grammar rules: This is where we declare the CFG rules for parsing along with the code segment that performs the routine that is required when a statement matches the grammar.

Subroutines: Used for wrting the main function where and other utility function in C. This is similar to the subroutine section in the lex file. This can also be used to import the lex.yy.c file to import the declared tokens.

Elements of LEX:

1. **yytext variable** : Whenevery a sequence of characters matches with a token, that text is held in the yytext variable. This text can then be indexed using the canonical format to extract the chracters.
2. **yylex()** : This indicates the main entry point for the lex. This function reads the input stream to generate the tokens and returns 0 at the end of the input stream.
3. **yywrap() :** This function is implicitly called by lex when the input stream ends. It returns 1 when the processing is done or 0 when more processing is required.
4. **yylval variable:** When a token is encountered, the value associated with the token is returned by lex in yyval variable. When we need to return the associated value with a token, we need to assign that value to the vvval varaible so that the YACC program will be able to access it.
5. **yyparse():** This function is used to initilaize parsing in yacc. It starts reading the input stream, identify the tokens, execute the routines and returns on EOF or an unrecoverable syntax error.
6. **yyerror():** This function is called when YACC encounters a grammar error while parsing the syntax. The purpose of this function is to display the error which caused the faileur in parsing.

Question 2

LEX code:

```
%{
%}

%%

"int" {return INT;}
"float" {return
FLOAT;}
([a-zA-Z])("_"|[a-zA-Z0-9])* {return
ID;} ";" {return SCOLON;}
"," {return
COMMA;} " " {}

%%
int yywrap()
{
	return 1;
}
```

YACC code:

```
%{
#include<stdio.h>
extern int yylex();
extern int yywrap();
extern int yyparse();
%}
%token ID INT FLOAT SCOLON COMMA

%%

D:	T L SCOLON{printf("Decleration
Statement");}; T:	INT |
	FLOAT ;
L:	L COMMA
	ID| ID;

%%

#include"lex.yy.c"
int yyerror(char
*str)
{
	printf("%s\n", str);
}
int main()
{
	yyparse();
}
```

Output:

```
a int;
syntax
error

float a,b;
Decleration Statement
```