Name: Sangeet Agrawal; PRN: 2107012214O; CS-B3

Practical No. 1 Date: 11/7/24

* Title: Theory assignment for writing details about LEX and YACC compilation.

* Objective: Learn about LEX and YACC compilation.

* LEX (Lexical Analyzer):

→ Purpose: Generates lexical analyzers to recognize patterns in text.

→ Components:

1) Definition Section
2) Rule Section
3) Auxiliary Functions

1) Definition Section: Includes declarations and regular definitions

→ Example:

```
%{ /* C code */ %}
letter [a-zA-Z]+
digit [0-9]+
```

2) Rule Section: Regular expressions & associated actions

→ Example:

```
{letter} {printf("Word: %s", yytext);}
```

3) Auxiliary Functions: User defined procedures and the main function.

→ Default main: `int main() { yylex(); return 0; }`

* Lex Process:
  → Input text is matched against regular expressions
  → Matched patterns trigger corresponding actions (eg. Returning a Token)
  → Unmatched patterns display an error.

* YACC (Yet Another Compiler Compiler)

  → Purpose: Generates LALR(1) parsers from grammar rules.
  → Components: 1) Definition Part
  2) Rule Part
  3) Auxiliary Routines Part

1) Definition Part: Includes token definitions and optional C code.

  → Example: %token NUMBER
  %token ID

2) Rule Part: Contains grammar definitions in a modified BNF form.

  → Example: expr: term '+' term { printf("Sum\n"); }

3) Auxiliary Routines Part: C code for functions used in the rules part.
  → Includes the main function if standalone:

```
int main() { yyparse(); return 0; }
```

* YACC Process:

1) Shift Action: Parser obtain the next token.
2) Reduce Action: Performs semantic actions based on grammar rules.
3) Output Files: y.tab.c (parser code),
y.tab.h (token definitions),
y.output (parsing table).

* Conclusion: Understanding LEX and YACC helps in creating lexical analyzers and parsers for various programming languages.

* Post-Lab Questions:

Q1) What are specifications for Regular Expressions?

Ans) Regular Expressions define patterns to match strings in text, eg: [a-zA-Z]+ matches any word consisting of letters.

Q2) Explain with an example section of LEX and YACC program.

Ans) LEX:-

```
%{ #include <stdio.h> %}
%%
[a-zA-Z]+ { printf("Word: %s", yytext); }
%%
int main() { yylex(); return 0; }
```

YACC:

```
%token NUMBER
%%
expr: NUMBER '+' NUMBER { printf("Sum\n"); }
%%
int main() { yyparse(); return 0; }
```

Q3) Explain Tokens with an example.

Ans) i) Tokens are the smallest units in a program, such as keywords, identifiers, or operators.

ii) Example: In the expression: 3+5,
the tokens are NUMBER(3), +,
NUMBER(5).

Q4) List commands for LEX and YACC program execution

Ans) LEX:

```
lex filename.l
cc lex.yy.c -o output
./output
```

YACC:

```
yacc -d filename.y
cc y.tab.c -o output
./output
```