**Practical – 7**

**Aim:**

**(a) Introduction to YACC.**

**(b) Write a YACC program to recognize language of all strings of the form anbn.**

**(a):**

- The yacc utility is a tool for writing compilers and other programs that parse input according to strict grammar rules.

- The yacc utility can produce anything from a simple parser for a desk calculator program to a very elaborate parser for a programming language.

- lex and yacc are a pair of programs that help write other programs. Input to lex and yacc describes how you want your final program to work.

- The output from lex is source code in the C programming language; you can compile this source code to get a program that works the way that you originally described.

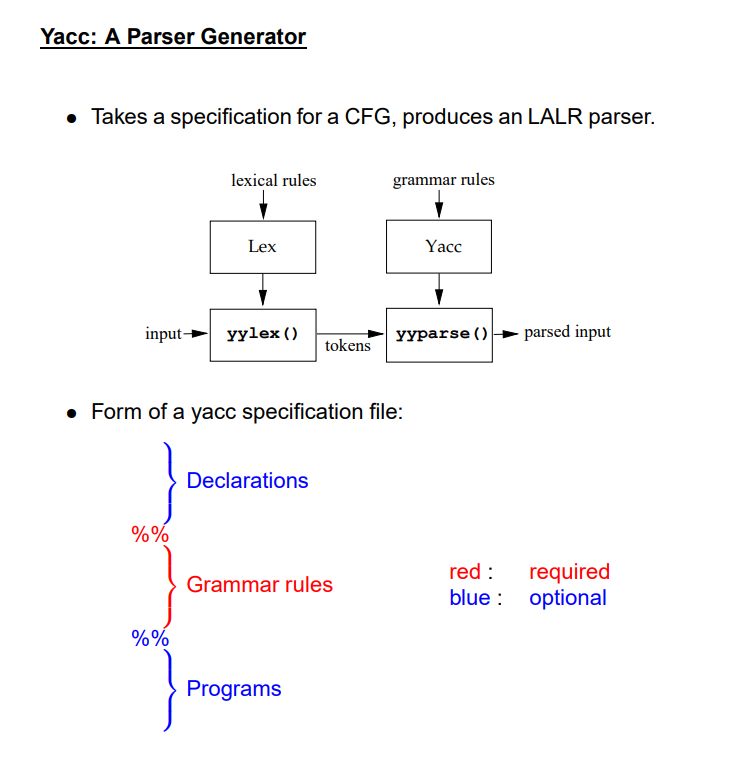
- The C code produced by lex analyzes input and breaks it into tokens. In the case of a simple desk calculator, math expressions must be divided into tokens.

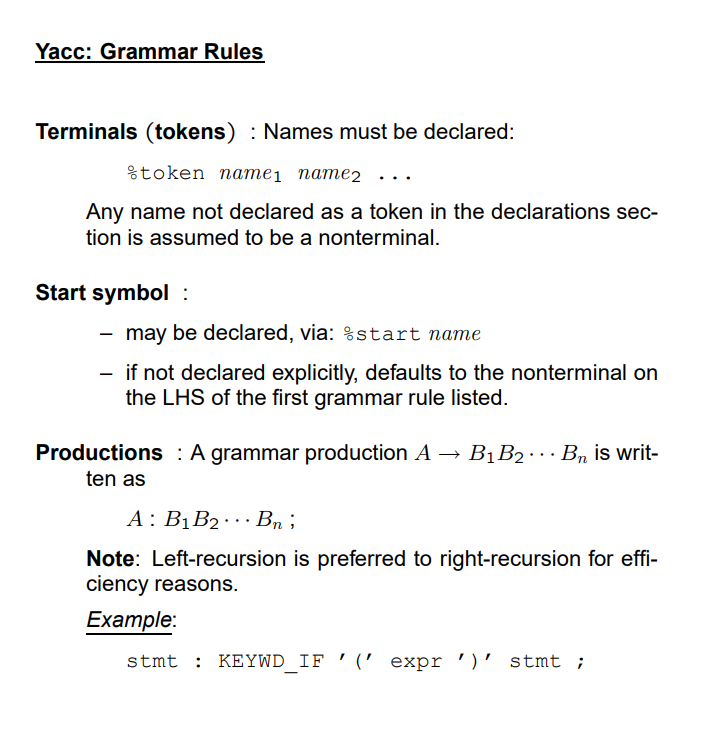
100 + 25 would be treated as 100, +, 25.

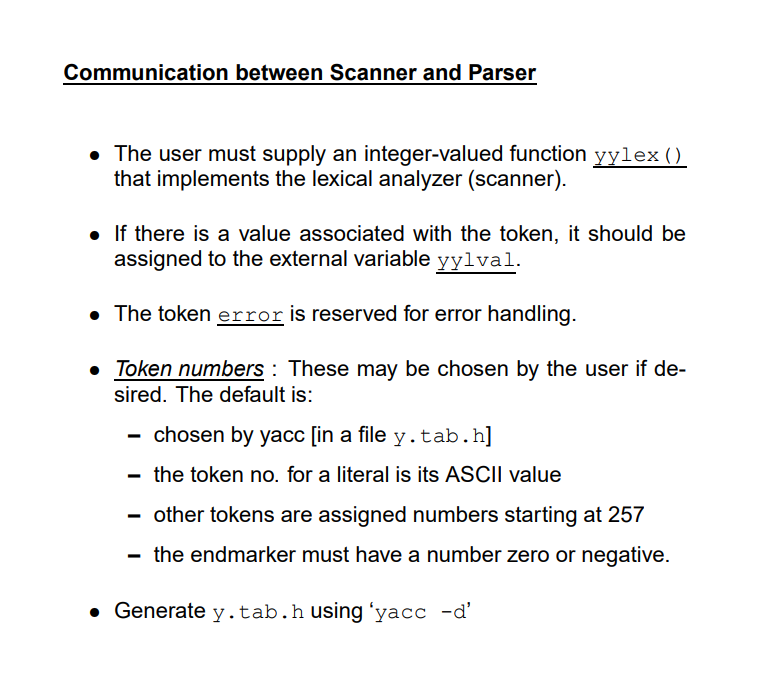
- The C code produced by yacc interprets the tokens that the lex code has obtained. For example, the yacc code figures out that a number followed by a + followed by another number means that you want to add the two numbers together.

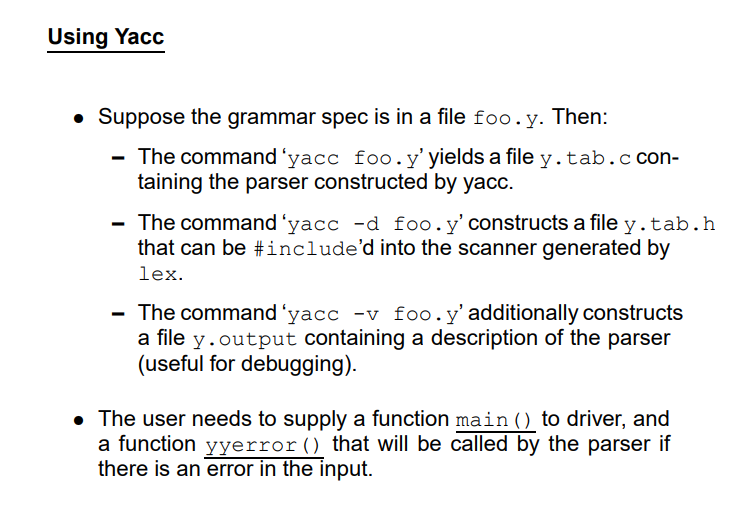
The goal of yacc is to generate the code for a C function named yyparse(). The yyparse() function calls yylex() to read a token from the standard input until end of file. yyparse() uses the return values from yylex() to figure out the types of each token obtained, and it uses yylval in each case as the actual value of that token.

The yyparse() function is called without any arguments. The result of the function is 0 if the input that was parsed was valid (that is, if the form of the input matched the descriptions given to lex and yacc). The result is 1 if the input contained errors of any kind.









**Code (b):**

**pr7b.l**

%{

#include "y.tab.h"

%}

%%

a return A;

b return B;

.|\n return yytext[0];

%%

**pr7b.y**

%{

#include <stdio.h>

void yyerror(char \*s);

int yylex(void);

%}

%token A B

%%

S1: S '\n' {

printf("string is valid\n");

return 0;

};

S : A S B|;

%%

void yyerror(char \*s) {

fprintf(stderr, "%s: detected\n", s);

}

int main() {

printf("Enter a string: ");

return yyparse();

}

**Output (b):**

