

# Department of CSE

## SSN College of Engineering

Vishakan Subramanian - 18 5001 196 - Semester VI

28 March 2021

---

### UCS 1602 - Compiler Design

---

#### Exercise 6: Implementation of Syntax Checker Using Yacc Tool

##### **Aim:**

Develop a **Syntax Checker** to recognize the tokens necessary for the following statements by writing suitable grammars.

- Assignment Statement
- Conditional Statement
- Looping Statement

## Code - Yacc Parser File:

```
1 %{
2     #include <stdio.h>
3     #define YYSTYPE double
4     int flag = 0;
5 %}
6
7 %token  NUM ASSIGN ID
8 %token  RELOP LOGIC ARITH INCDEC
9 %token  IF ELIF ELSE
10 %token  FOR WHILE
11
12 %%
13 Lines   :   Block Lines
14         |   Block
15         ;
16
17 Block   :   Stmt Block
18         |   Stmt
19         ;
20
21 Stmt    :   Loop '{' Block '}'
22         |   ConStmt '{' Block '}'
23         |   Expr ';'
24
25 Loop    :   FOR '(' Expr ';' Condns ';' Expr ')'
26         |   FOR '(' ';' Condns ';' ')'
27         |   WHILE '(' Condns ')'
28         ;
29
30 ConStmt :   IF '(' Condns ')'
31         |   ELIF '(' Condns ')'
32         |   ELSE
33         ;
34
35 Condns  :   Condns LOGIC Condns
36         |   Condns
37         ;
38
39 Condns  :   ID RELOP ID
40         |   ID RELOP NUM
41         |   ID
42         ;
43
44 Expr    :   Init
45         |   ID ASSIGN ID ARITH ID
46         |   ID ASSIGN ID ARITH NUM
47         |   ID ASSIGN NUM ARITH NUM
```

```

48         |      ID INCDEC
49         |      INCDEC ID
50         ;
51
52 Init      :      ID ASSIGN Init
53         |      ID ASSIGN ID
54         |      ID ASSIGN NUM
55         ;
56 %%
57
58 int yyerror(char *s){
59     flag = 1;
60     //fprintf(stderr, "%s\n", s);
61     return 1;
62 }
63
64 int main(void){
65     printf("\n\n\tSYNTAX CHECKER USING YACC\n");
66     printf("\nNote: Enter the code snippet in Code.txt.\n");
67     printf("\nCode Obtained:\n\n");
68     system("cat Code.txt");
69     yyparse();
70
71     if(flag){
72         printf("\nSyntactically Incorrect.\n");
73     }
74
75     else{
76         printf("\nSyntactically Correct.\n");
77     }
78
79     return 0;
80 }
81
82 /* Usage:
83     Run yacc -d Check.y
84     Run lex Check.l
85     Run gcc lex.yy.c -lm -w
86     Run ./a.out < Code.txt
87 */

```

## Code - Lex Grammar File:

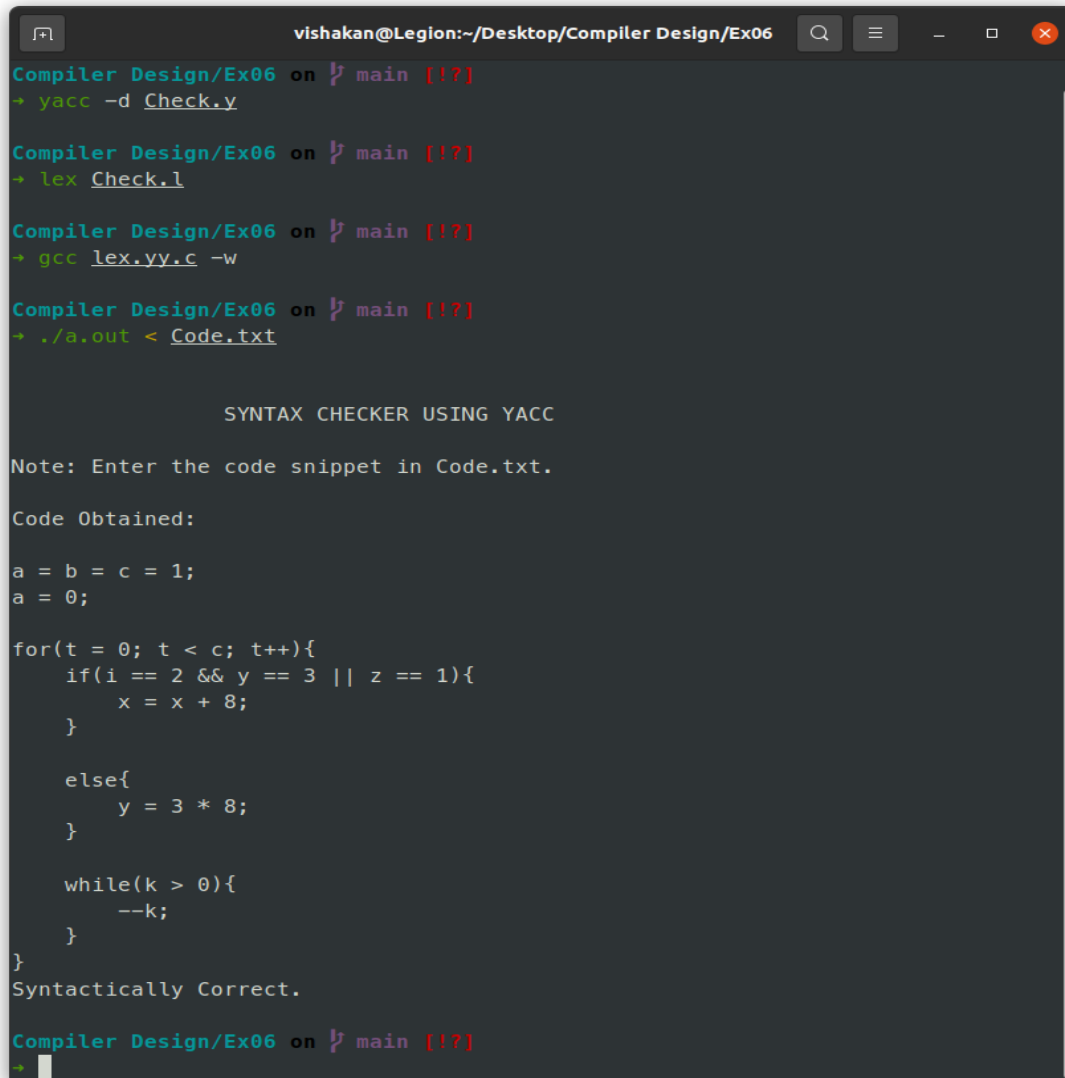
```
1 %{
2     #include <stdio.h>
3     #include "y.tab.c"
4     extern YYSTYPE yylval;
5 %}
6
7 assign      ("=")
8 relop       ("==" | "!=" | ">=" | "<=" | "<" | ">")
9 arithop     ("+" | "-" | "/" | "%" | "*")
10 incdec      ("++" | "--")
11 logical     ("||" | "&&")
12 identifier  [a-zA-Z_][a-zA-Z0-9_]*
13
14
15 %%
16
17 [0-9]+      {return NUM;}
18 {assign}    {return ASSIGN;}
19 {relop}     {return RELOP;}
20 {logical}   {return LOGIC;}
21 {arithop}   {return ARITH;}
22 {incdec}    {return INCDEC;}
23 "if"        {return IF;}
24 "else if"   {return ELIF;}
25 "else"      {return ELSE;}
26 "for"       {return FOR;}
27 "while"     {return WHILE;}
28 {identifier} {return ID;}
29
30
31 [ \t]       {;}
32 [\n]        {;}
33 .           {return *yytext;}
34
35 %%
36
37 int yywrap(){
38     return 1;
39 }
```

## Sample - Parsed C Code:

```
1 a = b = c = 1;
2 a = 0;
3
4 for(t = 0; t < c; t++){
5     if(i == 2 && y == 3 || z == 1){
6         x = x + 8;
7         a = b + c;
8     }
9
10    else{
11        y = 3 * 8;
12    }
13
14    while(k > 0){
15        --k;
16    }
17 }
```

## Output 1 - Valid Case:

Figure 1: Console Output - Valid Case.



```
vishakan@Legion:~/Desktop/Compiler Design/Ex06
Compiler Design/Ex06 on main [!?]
→ yacc -d Check.y

Compiler Design/Ex06 on main [!?]
→ lex Check.l

Compiler Design/Ex06 on main [!?]
→ gcc lex.yy.c -w

Compiler Design/Ex06 on main [!?]
→ ./a.out < Code.txt

SYNTAX CHECKER USING YACC

Note: Enter the code snippet in Code.txt.

Code Obtained:

a = b = c = 1;
a = 0;

for(t = 0; t < c; t++){
    if(i == 2 && y == 3 || z == 1){
        x = x + 8;
    }

    else{
        y = 3 * 8;
    }

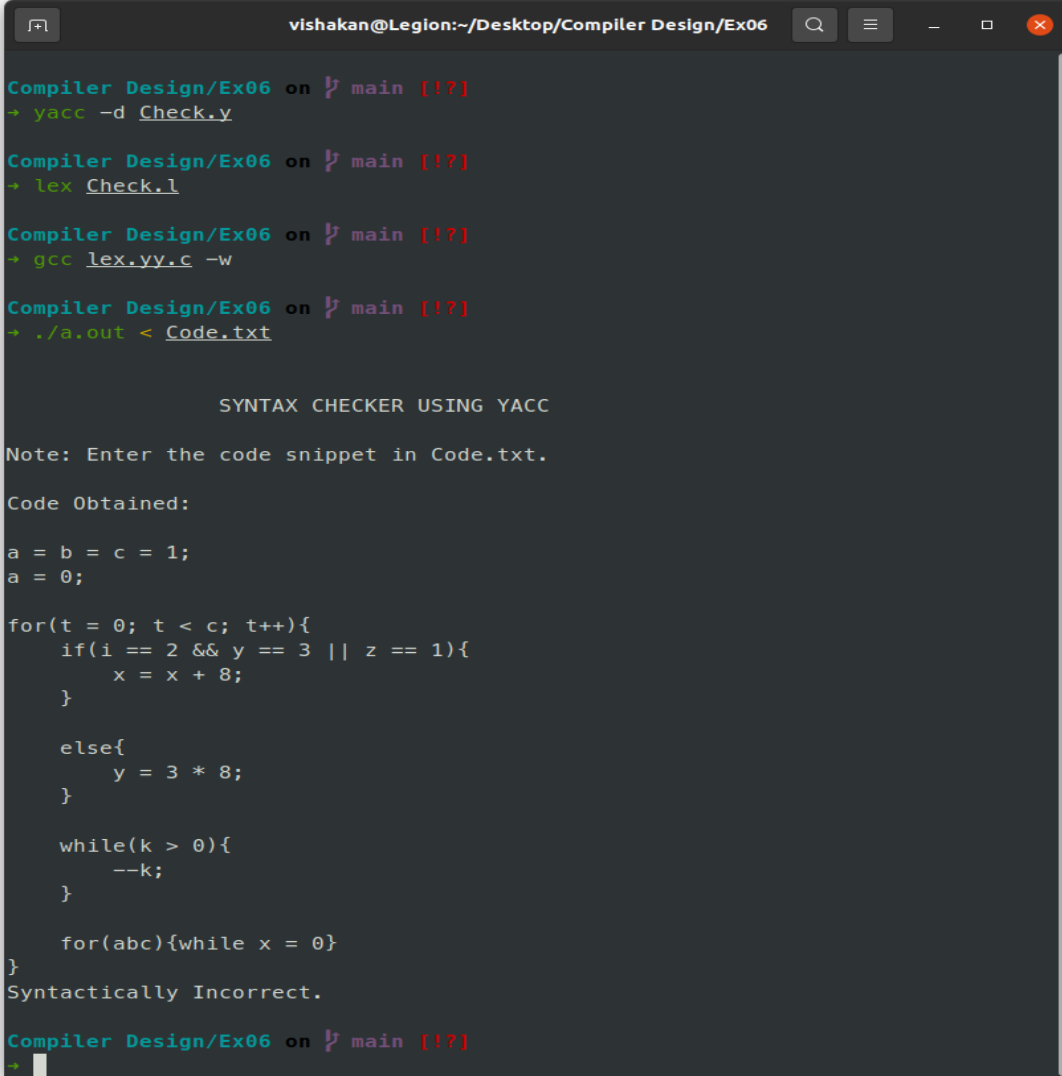
    while(k > 0){
        --k;
    }
}

Syntactically Correct.

Compiler Design/Ex06 on main [!?]
→
```

## Output 2 - Invalid Case:

Figure 2: Console Output - Invalid Case.



```
vishakan@Legion:~/Desktop/Compiler Design/Ex06
Compiler Design/Ex06 on  main [!?]
+ yacc -d Check.y
Compiler Design/Ex06 on  main [!?]
+ lex Check.l
Compiler Design/Ex06 on  main [!?]
+ gcc lex.yy.c -w
Compiler Design/Ex06 on  main [!?]
+ ./a.out < Code.txt

SYNTAX CHECKER USING YACC

Note: Enter the code snippet in Code.txt.

Code Obtained:

a = b = c = 1;
a = 0;

for(t = 0; t < c; t++){
    if(i == 2 && y == 3 || z == 1){
        x = x + 8;
    }

    else{
        y = 3 * 8;
    }

    while(k > 0){
        --k;
    }

    for(abc){while x = 0}
}
Syntactically Incorrect.

Compiler Design/Ex06 on  main [!?]
+ 
```

## Learning Outcome:

- I learnt more theory behind **Yacc Parser Generator**.
- I understood how to construct a grammar for a basic syntax checker.
- I learnt that grammar can be built upon layer by layer, each one adding more detail and complexity.
- I learnt that Yacc parser is able to handle Left Recursive grammar as well, since it is a LALR(1) parser.
- I was able to implement the required token recognition with Lex tool.
- I was able to implement a parser with Yacc to mimic the features of a syntax checker.
- I realized key implementation differences between the syntax checker and the desk calculator.
- I learnt how the Yacc parser catches an error using the inbuilt `yyerror()` function.