YACC

1) **Parser for IF-THEN Statements**

Program:

(if.l)

```
ALPHA [A-Za-z]
DIGIT [0-9]
%%
[ \t\n]
if                 return IF;
then                 return THEN;
{DIGIT}+             return NUM;
{ALPHA}({ALPHA}|{DIGIT})*          return ID;
"<="                 return LE;
">="                 return GE;
"=="                 return EQ;
"!="                 return NE;
"||"                 return OR;
"&&"                 return AND;
.                 return yytext[0];
%%
```

(if.y)

```
%{
#include <stdio.h>
#include <stdlib.h>
%}
%token ID NUM IF THEN LE GE EQ NE OR AND
%right '='
%left AND OR
%left '<' '>' LE GE EQ NE
%left '+''-'
%left '*''/'
%right UMINUS
%left '!'
```

```
%%
S   : ST {printf("Input accepted.\n");exit(0);};
ST  :    IF '(' E2 ')' THEN ST1';'
    ;
ST1 : ST
    | E
    ;
E   : ID'='E
    | E'+'E
    | E'-'E
    | E'*'E
    | E'/'E
    | E'<'E
    | E'>'E
    | E LE E
    | E GE E
    | E EQ E
    | E NE E
    | E OR E
    | E AND E
    | ID
    | NUM
    ;
E2  : E'<'E
    | E'>'E
    | E LE E
    | E GE E
    | E EQ E
    | E NE E
    | E OR E
    | E AND E
    | ID
    | NUM
    ;
%%

#include "lex.yy.c"

main()
```

```
{
printf("Enter the statement: ");
yyparse();
}
```

Output:

```
nn@linuxmint ~ $ lex if.l
nn@linuxmint ~ $ yacc if.y
nn@linuxmint ~ $ gcc y.tab.c -ll -ly
nn@linuxmint ~ $ ./a.out
Enter the statement: if(i>) then i=1;
syntax error
nn@linuxmint ~ $ ./a.out
Enter the statement: if(i>8) then i=1;
Input accepted.
nn@linuxmint ~ $
```

Parser for Switch Statements with If-then and While Statements inside

Program:

// **Lex file: pars.l**

```
alpha [a-zA-Z]
digit [0-9]

%%

[ \n\t]
if                  return IF;
then              return THEN;
while            return WHILE;
switch          return SWITCH;
case              return CASE;
default          return DEFAULT;
break            return BREAK;
{digit}+        return NUM;
```

```
{alpha}({alpha}|{digit})* return ID;
"<="            return LE;
">="            return GE;
"=="            return EQ;
"!="             return NE;
"&&"           return AND;
"||"              return OR;
.                  return yytext[0];

%%
```

// **Yacc file: pars.y**

```
%{
#include<stdio.h>
#include<stdlib.h>
%}

%token ID NUM SWITCH CASE DEFAULT BREAK LE GE EQ NE AND OR IF
THEN WHILE
%right '='
%left AND OR
%left '<' '>' LE GE EQ NE
%left '+''-'
%left '*''/'
%right UMINUS
%left '!'

%%

S    :    ST{printf("\nInput accepted.\n");exit(0);};
      ;
ST  :    SWITCH'('ID')''{'B'}'
      ;
B    :    C
      |    C D
      ;
C  :    C C
    |    CASE NUM':'ST1 BREAK';'
```

```
    ;
D  :    DEFAULT':'ST1 BREAK';'
    |    DEFAULT':'ST1
    ;
ST1    :    WHILE'('E2')' E';'
    |    IF'('E2')'THEN E';'
    |    ST1 ST1
    |    E';'
    ;
E2    :    E'<'E
    |    E'>'E
    |    E LE E
    |    E GE E
    |    E EQ E
    |    E NE E
    |    E AND E
    |    E OR E
    ;
E    :    ID'='E
    |    E'+'E
    |    E'-'E
    |    E'*'E
    |    E'/'E
    |    E'<'E
    |    E'>'E
    |    E LE E
    |    E GE E
    |    E EQ E
    |    E NE E
    |    E AND E
    |    E OR E
    |    ID
    |    NUM
    ;

%%

#include"lex.yy.c"
```

```
main()
{
    printf("\nEnter the expression: ");
    yyparse();
}
```

Output:

```
nn@linuxmint ~ $ lex pars.l
nn@linuxmint ~ $ yacc pars.y
nn@linuxmint ~ $ gcc y.tab.c -ll -ly
nn@linuxmint ~ $ ./a.out

Enter the expression: switch(s)
{
case 1:a=b+c;break;
case 2:    if(a<10)
    then a=b*c;
    break;
case 3:    while(a<5)
    b=b+a;
    break;
}

Input accepted.
nn@linuxmint ~ $
```

Postfix to Infix - Yacc Program - Compiler Design

Program:-"Beta version"-(Partial Output Only)

// **Lex file: ptoi.l**

```
DIGIT [0-9]
%%
{DIGIT}+    {yylval=atoi(yytext);return ID;}
```

```
[-+*/]            {return yytext[0];}
.                      ;
\n         yyterminate();
```

// **Yacc file: ptoi.y**

```
%{
    #include<stdio.h>
    #include<string.h>
    void push();
    char* top();
    void a1(char* a);
%}

%token ID

%%

S    : E  { printf("= %s \n",top());}
    ;
E    : E E '+' {a1(" + ");}
    | E E '*' {a1(" * ");}
    | E E '-' {a1(" - ");}
    | E E '/' {a1(" / ");}
    | ID    {push();}
    ;

%%
#include"lex.yy.c"

char st[100][10];
int indx=0;

void push()
{
   strcpy(st[indx++],yytext);
}

char* pop()
{
```

```
    return st[--indx];
}

char* top()
{
    return st[indx-1];
}
void a1(char* a)
{
    char buffer[20];
    char* c1=pop();
    char* c2=pop();
    bzero(buffer,20);
    strcat(buffer,c2);
    strcat(buffer,a);
    strcat(buffer,c1);
    strcpy(st[indx++],buffer);
}
main()
{
    yyparse();

}
```

Output:

```
nn@linuxmint ~ $ lex ptoi.l
nn@linuxmint ~ $ yacc ptoi2.y
nn@linuxmint ~ $ gcc y.tab.c -ll -ly
nn@linuxmint ~ $ ./a.out
2 5 * 3 2 * +
= 2 * 5 + 3 * 2
nn@linuxmint ~ $ ./a.out
2 5 +
= 2 + 5
nn@linuxmint ~ $ ./a.out
2 3 *
= 2 * 3
nn@linuxmint ~ $ ./a.out
2 3 -
```

```
= 2 - 3
nn@linuxmint ~ $
```