AFLL ASSIGNMENT 2

Name 1 : Nishanth D’Mello
SRN 1 : PES2UG21CS343

Name 2 : N Digvijay
SRN 2 : PES2UG21CS310

Sem : 3rd
Section : F

1) for loop in C++ language

Grammar :

S->FOR LPAREN DATA RPAREN LCURLY STAT RCURLY
DATA->A SEMI B SEMI C
A->ID EQUAL ID | ID EQUAL NUM | NULL
B->ID COMPARE ID | NUM COMPARE NUM | ID COMPARE NUM | NUM COMPARE ID | ID | NUM | NULL
COMPARE->LESSER | LESSER EQUAL | GREATER | GREATER EQUAL | NOT EQUAL | EQUAL EQUAL
C->ID EQUAL ID SYMB ID | ID EQUAL NUM SYMB ID | ID EQUAL ID SYMB NUM | ID EQUAL NUM SYMB NUM | NULL
SYMB->PLUS | MINUS | STAR | DIVIDE | MOD
STAT->PRINT SEMI STAT | C SEMI STAT | PRINT SEMI | C SEMI
PRINT->COUT X
X->LESSER LESSER ID | LESSER LESSER NUM | LESSER LESSER ID X | LESSER LESSER NUM X | LESSER LESSER QUOTE ID QUOTE | LESSER LESSER QUOTE ID QUOTE X

Python Code :

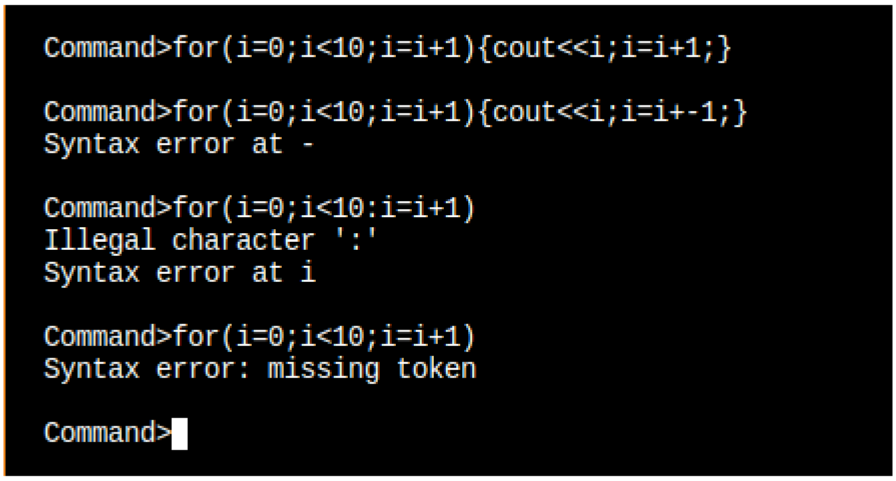
```
import ply.lex as lex
import ply.yacc as yacc

reserved = {"for": "FOR", "cout": "COUT"}

tokens = list(reserved.values()) + [“LPAREN","ID","EQUAL","NUM","SEMI","LESSER","GREATER",
"NOT","PLUS","STAR","MINUS","DIVIDE","MOD","RPAREN","LCURLY","RCURLY","QUOTE","NULL",]

t_PLUS = r"\+"
t_MINUS = r"\-"
t_STAR = r"\*"
t_DIVIDE = r"\/"
t_MOD = r"\%"
t_NOT = r"\!"
t_EQUAL = r"\="
t_SEMI = r"\;"
t_LPAREN = r"\("
t_RPAREN = r"\)"
t_ignore = "\t"
t_LESSER = r"\<"
t_GREATER = r"\>"
t_LCURLY = r"\{"
t_RCURLY = r"\}"
```

```
t_NULL = r"\ "
t_QUOTE = r"\""

def t_NUM(t):
r"[0-9]+"
t.type = reserved.get(t.value, "NUM")
return t

def t_ID(t):
r"[a-zA-Z]+"
t.type = reserved.get(t.value, "ID")
return t

def t_error(t):
print("Illegal character '%s'" % t.value[0])
t.lexer.skip(1)

lexer = lex.lex()

def p_1(p):
"S : FOR LPAREN DATA RPAREN LCURLY STAT RCURLY"

def p_2(p):
"DATA : A SEMI B SEMI C"

def p_3(p):
"A : ID EQUAL NUM"

def p_4(p):
"A : ID EQUAL ID"

def p_5(p):
"A : NULL"

def p_6(p):
"B : ID COMPARE ID"

def p_7(p):
"B : ID COMPARE NUM"

def p_8(p):
"B : NUM COMPARE ID"

def p_9(p):
"B : NUM COMPARE NUM"

def p_10(p):
"B : NULL"

def p_11(p):
"B : ID"
```

```
def p_12(p):
"C : ID EQUAL ID SYMB NUM"

def p_13(p):
"C : ID EQUAL ID SYMB ID"

def p_14(p):
"C : ID EQUAL NUM SYMB NUM"

def p_15(p):
"C : ID EQUAL NUM SYMB ID"

def p_16(p):
"C : NULL"

def p_17(p):
"SYMB : PLUS"

def p_18(p):
"SYMB : MINUS"

def p_19(p):
"SYMB : STAR"

def p_20(p):
"SYMB : DIVIDE"

def p_21(p):
"SYMB : MOD"

def p_22(p):
"COMPARE : LESSER"

def p_23(p):
"COMPARE : GREATER"

def p_24(p):
"COMPARE : LESSER EQUAL"

def p_25(p):
"COMPARE : GREATER EQUAL"

def p_26(p):
"COMPARE : EQUAL EQUAL"

def p_27(p):
"COMPARE : NOT EQUAL"

def p_28(p):
"STAT : PRINT SEMI STAT"
```

```
def p_29(p):
"STAT : C SEMI STAT"

def p_30(p):
"STAT : C SEMI"

def p_31(p):
"STAT : PRINT SEMI"

def p_32(p):
"PRINT : COUT X"

def p_33(p):
"X : LESSER LESSER ID"

def p_34(p):
"X : LESSER LESSER ID X"

def p_35(p):
"X : LESSER LESSER NUM"

def p_36(p):
"X : LESSER LESSER NUM X"

def p_37(p):
"X : LESSER LESSER QUOTE ID QUOTE"

def p_38(p):
"X : LESSER LESSER QUOTE ID QUOTE X"

def p_error(t):
if t:
print("Syntax error at %s" % t.value)
else:
print("Syntax error: missing token")

parser = yacc.yacc()

while True:
try:
S = input("\nCommand>")
if S == "q":
print()
break
except EOFError:
break
parser.parse(S)
```

Output :



```
Command>for(i=0;i<10;i=i+1){cout<<i;i=i+1;}

Command>for(i=0;i<10;i=i+1){cout<<i;i=i+-1;}
Syntax error at -

Command>for(i=0;i<10:i=i+1)
Illegal character ':'
Syntax error at i

Command>for(i=0;i<10;i=i+1)
Syntax error: missing token

Command>
```

2)while loop in C++ language

Grammar :

S->WHILE LPAREN COND RPAREN LCURLY STAT RCURLY
COND->ID COMPARE ID | NUM COMPARE NUM | ID COMPARE NUM | NUM COMPARE ID | ID | NUM
COMPARE->LESSER | LESSER EQUAL | GREATER | GREATER EQUAL | NOT EQUAL | EQUAL EQUAL
STAT->PRINT SEMI STAT | UPD SEMI STAT | PRINT SEMI | UPD SEMI
PRINT->COUT X
X->LESSER LESSER ID | LESSER LESSER NUM | LESSER LESSER ID X | LESSER LESSER NUM X | LESSER LESSER QUOTE ID QUOTE | LESSER LESSER QUOTE ID QUOTE X
UPD->ID EQUAL ID SYMB ID | ID EQUAL NUM SYMB ID | ID EQUAL ID SYMB NUM | ID EQUAL NUM SYMB NUM
SYMB->PLUS | MINUS | STAR | DIVIDE | MOD

Python Code :

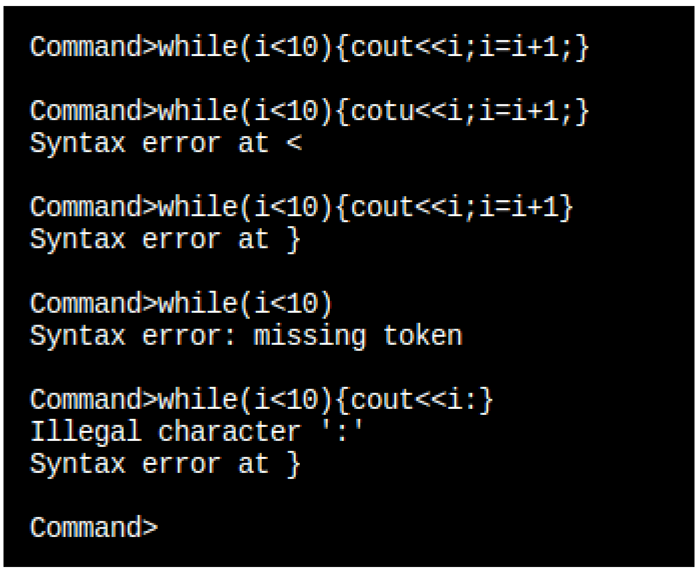
```
import ply.lex as lex
import ply.yacc as yacc


reserved = {"while": "WHILE", "cout": "COUT"}


tokens = list(reserved.values()) + ["LPAREN","ID","EQUAL","NUM","SEMI","LESSER","GREATER","NOT",
"PLUS","STAR","MINUS","DIVIDE","MOD","RPAREN","LCURLY","RCURLY","QUOTE",]


t_PLUS = r"\+"
t_MINUS = r"\-"
t_STAR = r"\*"
t_DIVIDE = r"\/"
t_MOD = r"\%"
t_NOT = r"\!"
t_EQUAL = r"\="
t_SEMI = r"\;"
t_LPAREN = r"\("
t_RPAREN = r"\)"
t_ignore = "\t"
t_LESSER = r"\<"
t_GREATER = r"\>"
t_LCURLY = r"\{"
t_RCURLY = r"\}"
```

```
t_QUOTE = r"\""

def t_NUM(t):
r"[0-9]+"
t.type = reserved.get(t.value, "NUM")
return t

def t_ID(t):
r"[a-zA-Z]+"
t.type = reserved.get(t.value, "ID")
return t

def t_error(t):
print("Illegal character '%s'" % t.value[0])
t.lexer.skip(1)

lexer = lex.lex()

def p_1(p):
"S : WHILE LPAREN COND RPAREN LCURLY STAT RCURLY"

def p_2(p):
"COND : ID COMPARE ID"

def p_3(p):
"COND : ID COMPARE NUM"

def p_4(p):
"COND : NUM COMPARE ID"

def p_5(p):
"COND : NUM COMPARE NUM"

def p_6(p):
"COND : ID"

def p_7(p):
"COND : NUM"

def p_8(p):
"COMPARE : LESSER"

def p_9(p):
"COMPARE : GREATER"

def p_10(p):
"COMPARE : LESSER EQUAL"

def p_11(p):
"COMPARE : GREATER EQUAL"
```

```
def p_12(p):
"COMPARE : EQUAL EQUAL"

def p_13(p):
"COMPARE : NOT EQUAL"

def p_14(p):
"STAT : PRINT SEMI STAT"

def p_15(p):
"STAT : UPD SEMI STAT"

def p_16(p):
"STAT : UPD SEMI"

def p_17(p):
"STAT : PRINT SEMI"

def p_18(p):
"UPD : ID EQUAL ID SYMB ID"

def p_19(p):
"UPD : ID EQUAL NUM SYMB ID"

def p_20(p):
"UPD : ID EQUAL NUM SYMB NUM"

def p_21(p):
"UPD : ID EQUAL ID SYMB NUM"

def p_22(p):
"SYMB : PLUS"

def p_23(p):
"SYMB : MINUS"

def p_24(p):
"SYMB : STAR"

def p_25(p):
"SYMB : DIVIDE"

def p_26(p):
"SYMB : MOD"

def p_27(p):
"PRINT : COUT X"

def p_28(p):
"X : LESSER LESSER ID"
```

```
def p_29(p):
"X : LESSER LESSER ID X"

def p_30(p):
"X : LESSER LESSER NUM"

def p_31(p):
"X : LESSER LESSER NUM X"

def p_32(p):
"X : LESSER LESSER QUOTE ID QUOTE"

def p_33(p):
"X : LESSER LESSER QUOTE ID QUOTE X"

def p_error(t):
if t:
print("Syntax error at %s" % t.value)
else:
print("Syntax error: missing token")

parser = yacc.yacc()

while True:
try:
S = input("\nCommand>")
if S == "q":
print()
break
except EOFError:
break
parser.parse(S)
```

Output :



```
Command>while(i<10){cout<<i;i=i+1;}

Command>while(i<10){cotu<<i;i=i+1;}
Syntax error at <

Command>while(i<10){cout<<i;i=i+1}
Syntax error at }

Command>while(i<10)
Syntax error: missing token

Command>while(i<10){cout<<i:}
Illegal character ':'
Syntax error at }

Command>
```