Yacc Is a tool that takes a context-free grammar as input and produces a parser for that grammar as output.

CONFLICT : conflict means that during the course of generating a parser for the input grammar, yacc faced a choice between multiple possible actions for the parser to take.

Parsers repeatedly perform two actions:shift and reduce. The general idea of the parsing algorithm is to match the input that has been seen with the right hand sides of grammar productions until an appropriate match is found, to replace it with the corresponding left hand side. Parser stores the grammar symbols it has found so far on its stack. A shift action takes the next token from the input and pushes it onto the stack. A Reduce Action happens when the sequence of grammar symbols on top of the parser's stack matches the right hand side and the parser decides to reduce this to the left hand side of the grammar rule.

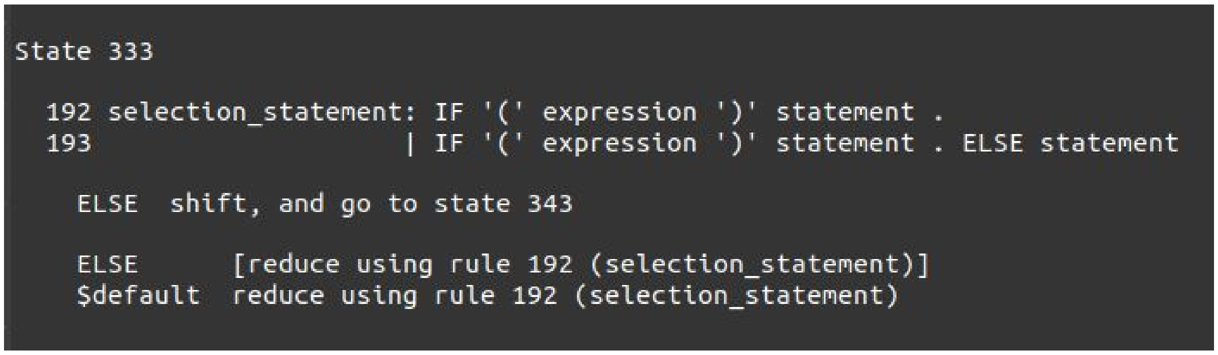
There are two types of conflicts we might encounter:
1.Shift-reduce conflict:This occurs when the parser is faced with a choice of a shift action and a reduce action.
2.Reduce-reduce conflict.This occurs when the parser is faced with a choice of two different productions that could be used for a reduce action.

Conflict Resolution:
Yacc's default action
- in the case of a shift-reduce conflict is to choose the shift action.
- in the case of a reduce-reduce conflict is to reduce using the production that comes first in grammar specification.

For the given files :
1.CGrammar-y.txt
To get detailed output from yacc use the below command in terminal:
```
$ yacc -d -v CGrammar-y.txt
```
A file named 'y.output' is produced in the same directory as a result of the above command. We can see all conflicts obtained in the beginning of this file.

Obtained conflicts for CGrammar-y.txt:

State 333 conflicts: 1 shift/reduce



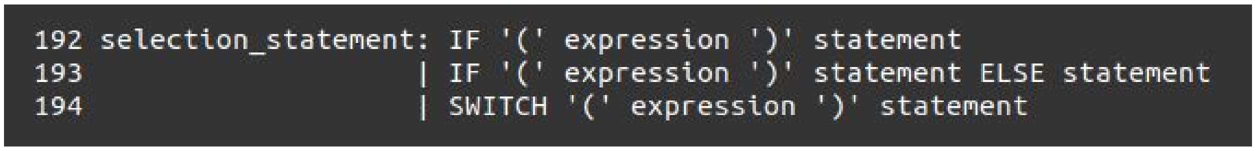
```
State 333

  192 selection_statement: IF '(' expression ')' statement .
  193                    | IF '(' expression ')' statement . ELSE statement

    ELSE  shift, and go to state 343

    ELSE      [reduce using rule 192 (selection_statement)]
    $default  reduce using rule 192 (selection_statement)
```

192,193 are the rule numbers in the given grammar.



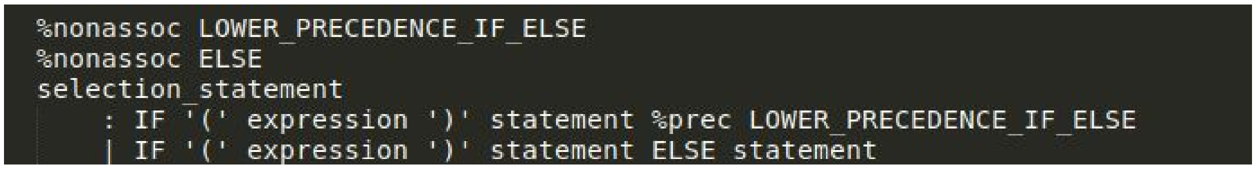
```
192 selection_statement: IF '(' expression ')' statement
193                    | IF '(' expression ')' statement ELSE statement
194                    | SWITCH '(' expression ')' statement
```

“$default reduce using rule 192 (selection_statement)” in state 333 means that by default current state reduces by rule 192.
The introduction of the ELSE token gives the parser two choices:

- Either it can be reduced using rule 192 or
- It can shift and scan ELSE token and go to state 343.

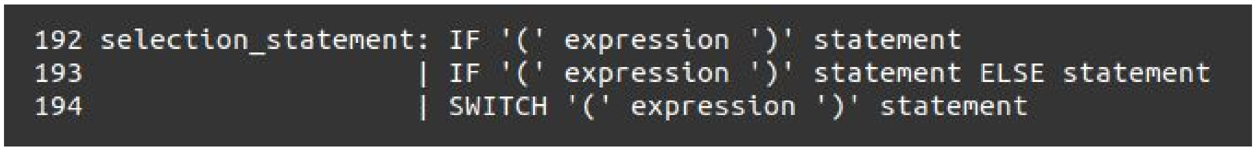
Since yacc prioritizes shift over reduce, the ELSE is scanned and the parser goes to state 343.

We can resolve this conflict by
1.Assigning precedence to rules that cause conflict.
Make the following changes to rule 192,193.
We introduce a new pseudo token ‘LOWER_PRECEDENCE_IF_ELSE’ and make this token and ELSE token non-associative. Higher precedence is given to shift over reduce and this resolves the conflict.



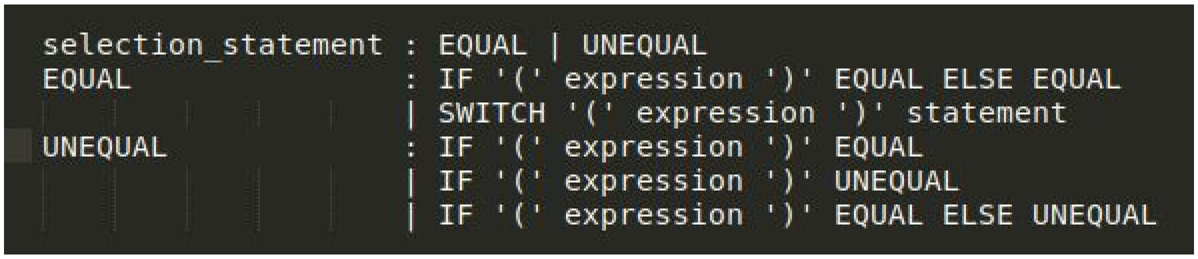
```
%nonassoc LOWER_PRECEDENCE_IF_ELSE
%nonassoc ELSE
selection_statement
    : IF '(' expression ')' statement %prec LOWER_PRECEDENCE_IF_ELSE
    | IF '(' expression ')' statement ELSE statement
```

2.Restructuring the ambiguous grammar.



```
selection_statement: IF '(' expression ')' statement
                   | IF '(' expression ')' statement ELSE statement
                   | SWITCH '(' expression ')' statement
```

Denote SWITCH ‘(‘ expression ‘)’ by S to keep the representation short.
Consider case
IF(expression) if(expression) S else S
This can be interpreted as IF(expression)[IF(expression) S else S] (case 1)or IF(expression)[IF(expression) S] else S.(case 2) We can fix this ambiguity by restructuring the grammar.We consider case1 to be the expected expression.



```
selection_statement : EQUAL | UNEQUAL
EQUAL               : IF '(' expression ')' EQUAL ELSE EQUAL
                    | SWITCH '(' expression ')' statement
UNEQUAL             : IF '(' expression ')' EQUAL
                    | IF '(' expression ')' UNEQUAL
                    | IF '(' expression ')' EQUAL ELSE UNEQUAL
```

Here EQUAL matches statements with equal number of if and else and those without equal number of if and else are matched by UNEQUAL.

These above 2 methods resolve the conflicts.

2.CxxGrammar-y.txt:
To get detailed output from yacc use the below command in terminal:
$ yacc -d -v CxxGrammar-y.txt
A file named ‘y.output’ is produced in the same directory as a result of the above command.
We don’t get any conflicts from the CxxGrammar file. But the following lines mentioned the following in the grammar.

“ The C++ grammar comprises 558 rules and uses 894 states in yacc, with 0 unresolved conflicts. 24 conflicts from 10 ambiguities are resolved by 8 %prec's, so that yacc and bison report 0 conflicts. “ All the conflicts are already resolved using yacc precedence.