CS3423 “Compilers-II”
Mini-Assignment #3 Report

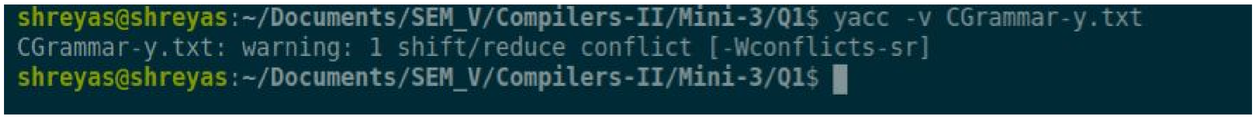
Shreyas Jayant Havaldar
CS18BTECH11042

Question 1

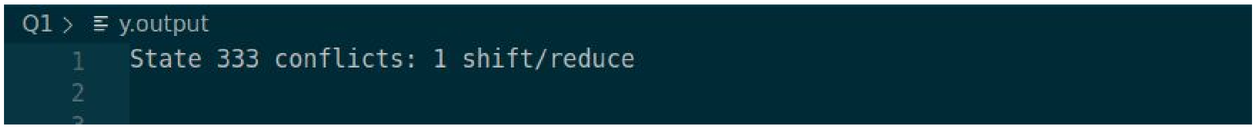
C:

Conflicts Found:

One shift-reduce conflict is found in the grammar given to us.



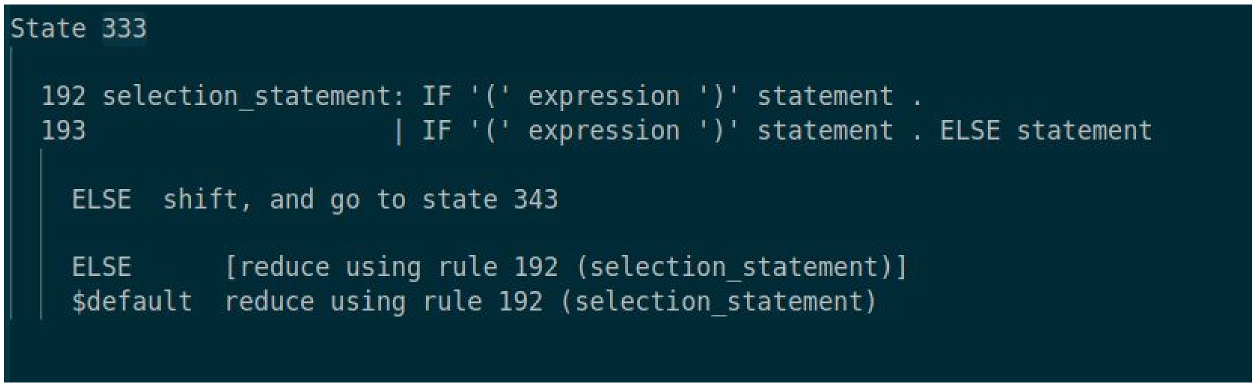
```
shreyas@shreyas:~/Documents/SEM_V/Compilers-II/Mini-3/Q1$ yacc -v CGrammar-y.txt
CGrammar-y.txt: warning: 1 shift/reduce conflict [-Wconflicts-sr]
shreyas@shreyas:~/Documents/SEM_V/Compilers-II/Mini-3/Q1$
```



Q1 > y.output

```
State 333 conflicts: 1 shift/reduce
```

In the *y.output* file, the details are provided and we note that the conflict is in state **333**.



```
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)
```

As expected this is a shift-reduce conflict in the nested if-else conditional statement, the classic "dangling else" problem.

Understanding the Cause:

We have an issue as we need to decide if we should shift the **ELSE** or reduce the **IF expression statement** at the top of the stack.

If we shift first then we have a different pairing of the IF and ELSE constructs than we would have if we reduce.

In the first case, the latest ELSE is paired with the most recent unpaired IF, innermost nesting.

In the second case, the latter ELSE is paired with the first IF, the outer IF.

Thus **yacc** gives us a warning message so that we are aware of the ambiguity in the grammar although its default behavior, when a shift-reduce conflict is encountered, is to shift which is generally followed.

Resolution:

We usually pair an **ELSE** keyword with the most recent unpaired **IF.** We create a pseudo-token LESSER_ELSE. So we simply resolve this conflict by giving **IF-ELSE** higher precedence than the simple **IF** statement. Thus we are choosing to shift the ELSE as is the norm.

We utilise the pseudo-token and our ELSE token and make them both non-associative:

```
%nonassoc LESSER_ELSE

%nonassoc ELSE
```

This gives ELSE more precedence over LESSER_ELSE simply because LESSER_ELSE is declared first. The modified rule for the resolved state 333 is:

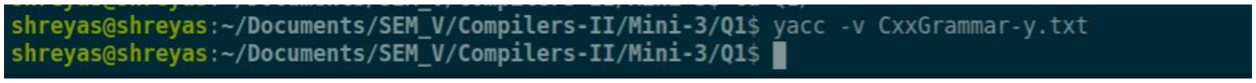
```
selection_stmt :   IF  '(' expression ')' statement    %prec LESSER_ELSE ;

                   | IF '(' expression ')' statement ELSE statement ;
```

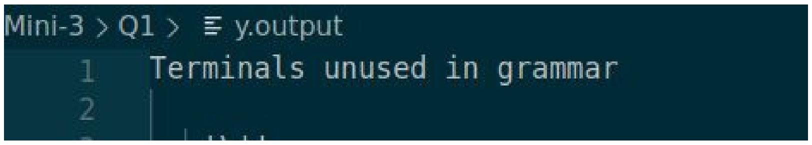
C++:

Conflicts Found:

No conflicts found in the grammar given to us.



```
shreyas@shreyas:~/Documents/SEM_V/Compilers-II/Mini-3/Q1$ yacc -v CxxGrammar-y.txt
shreyas@shreyas:~/Documents/SEM_V/Compilers-II/Mini-3/Q1$
```



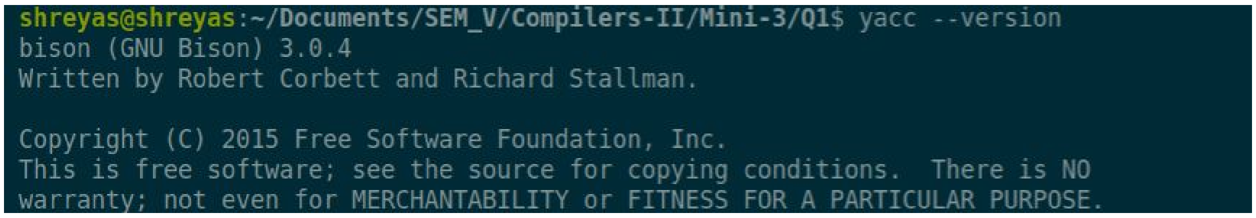
```
Terminals unused in grammar
```

In the *y.output* file as well we can find that there is indeed no conflict and the grammar is conflict-free!

Question 2

Note: Refer the README.md and lex and yacc program files for details.

Note the version details.



```
shreyas@shreyas:~/Documents/SEM_V/Compilers-II/Mini-3/Q1$ yacc --version
bison (GNU Bison) 3.0.4
Written by Robert Corbett and Richard Stallman.

Copyright (C) 2015 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```