Name: Shardul Vikram Dharmadhikari
Roll no: 8016

ASSIGNMENT_7

Problem Definition:

Write a program using YACC specifications to implement syntax analysis phase of compiler to recognize simple and compound sentences given in input file.

1.1 Prerequisite:

Basic concepts of phaes of complier(syntax analyzer)

1.2 Learning Objectives:

Understand the implementation Yacc Specification with simple & compound sentences

1.3 Syntax Analyzer:-

Syntax Analysis or Parsing is the second phase, i.e. after lexical analysis. It checks the syntactical structure of the given input, i.e. whether the given input is in the correct syntax (of the language in which the input has been written) or not. It does so by building a data structure, called a Parse tree or Syntax tree. The parse tree is constructed by using the pre-defined Grammar of the language and the input string. **If the given input string can be produced with the help of the syntax tree (in the derivation process), the input string is found to be in the correct syntax.**

Example:

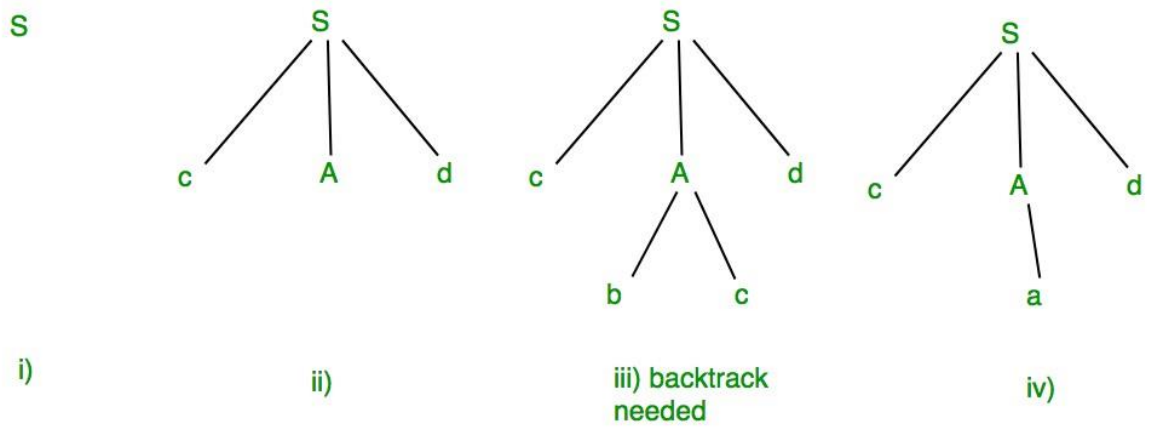
Suppose Production rules for the Grammar of a language are:

```
S -> cAd

A -> bc|a

And the input string is “cad”.
```

Now the parser attempts to construct syntax tree from this grammar for the given input string. It uses the given production rules and applies those as needed to generate the string. To generate string “cad” it uses the rules as shown in the given diagram:



1.4 Parse Tree:-

Parse tree is a hierarchical structure which represents the derivation of the grammar to yield input strings.

A parse tree is an entity which represents the structure of the derivation of a terminal string from some non-terminal (not necessarily the start symbol). The definition is as in the book. Key features to define are the *root* $\in V$ and *yield* $\in \Sigma^*$ of each tree

- For each $\sigma \in \Sigma$, there is a tree with root σ and no children; its yield is σ
- For each rule $A \rightarrow \varepsilon$, there is a tree with root A and one child ε; its yield is ε
- If $t_1, t_2, ..., t_n$ are parse trees with roots $r_1, r_2, ..., r_n$ and respective yields $y_1, y_2, ..., y_n$, and $A \rightarrow r_1 r_2 ... r_n$ is a production, then there is a parse tree with root A whose children are $t_1, t_2, ..., t_n$. Its root is A and its yield is $y_1 y_2 ... y_n$

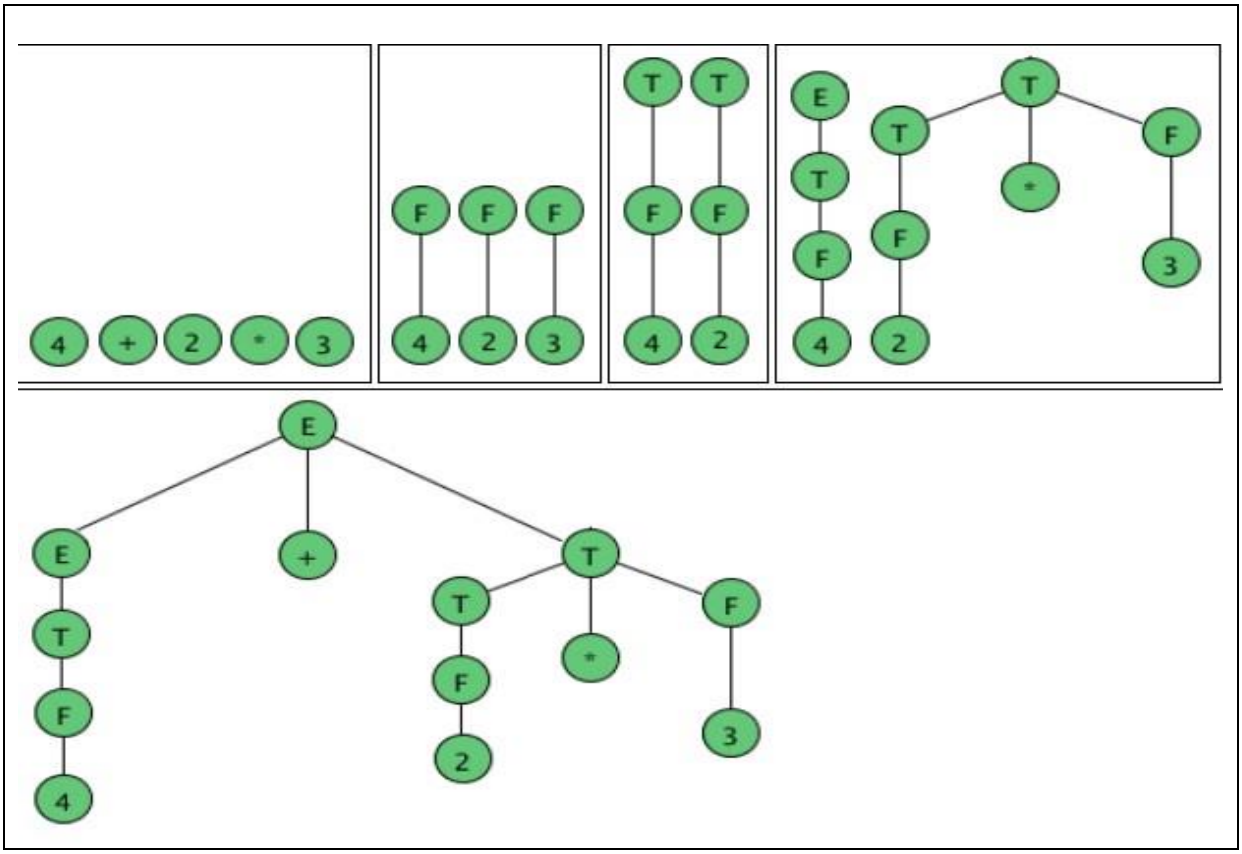
Observe that parse trees are constructed from **bottom up**, not top down. The actual construction of "adding children" should be made more precise, but we intuitively know what's going on.
As an example, here are all the parse (sub) trees used to build the parse tree for the arithmetic expression 4 + 3 * 2 using the expression grammar

$E \rightarrow E + T \mid E - T \mid T$

$T \rightarrow T * F \mid F$

$F \rightarrow \mathbf{a} \mid (E)$

Where **a** represents an operand of some type, be it a number or variable.
The trees are grouped by height.



Code:

```
%{
#include<stdio.h>
#include "y.tab.h"
%}
%%
int|float|char|boolean     {return DATATYPE;}
[a-zA-Z] {return IDENTIFIER;}
"," {return COMMA;}
";" {return SEMICOLON;}
":" {return COLON;}
[" "] {return SPACE;}
private|protected|default|public {return ACCESS;}
%%
int yywrap()
{
```

```
return 1;
}
```