Q1. Developing a simple calculator

Q.1.1. Describe Context Free Grammar (CFG) for arithmetic expressions. Operations to consider are addition, subtraction, division and multiplication. Check that your grammar Is suitable for LR parsing.

The Grammar used is as follows:

S -> E

E -> E + T | E – T | T

T -> T * F | T / F | F

F -> (E) | number

The grammar is ambiguous but the ambiguity can be removed with one look ahead symbol. Hence it is suitable for LR(1) parsing.

Q.1.2. Design an attribute grammar (by adding appropriate attributes and semantic rules) for the CFG developed in 1.1 to be able to evaluate the result of given arithmetic expression. Consider an example expression and illustrate how your solution works (using parse tree, dependency graph).

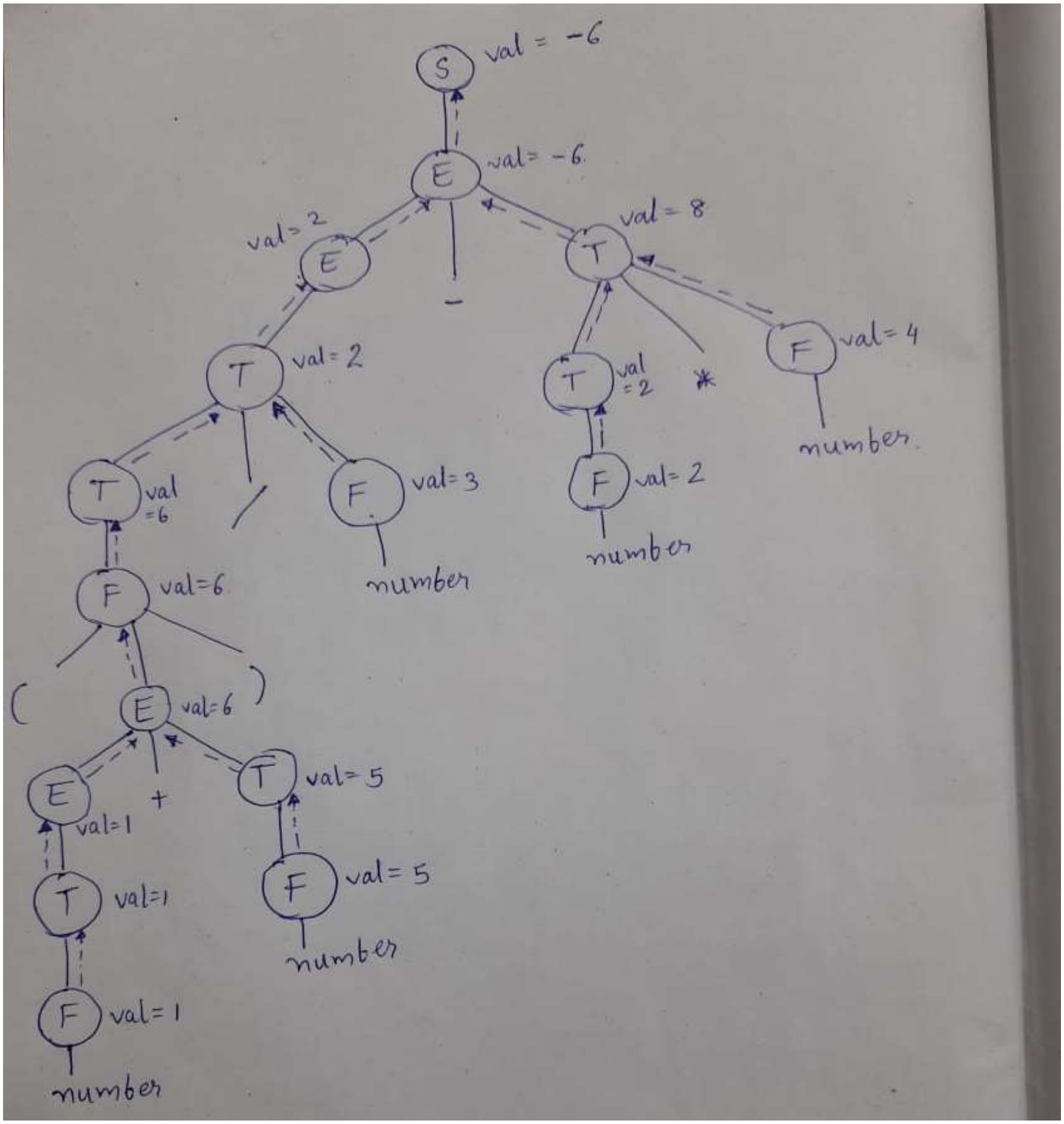
The Attribute grammar for above CFG will be as follows:

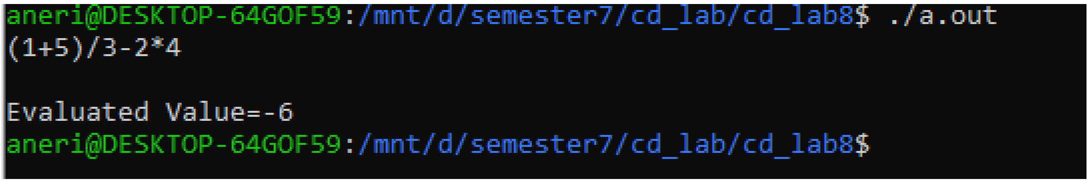
S -> E	{S.value (↑)=E.value(↑)}
E 1 -> E2 + T	{E1.value (↑) = E2.value (↑)+ T.value(↑) }
-> E2 - T	{E1.value (↑) = E2.value (↑)- T.value(↑) }
-> T	{E1.value (↑) = T.value (↑) }
T1 -> T2 * F	{T1.value (↑) = T2.value (↑)*F.value (↑) }
->T2 / F	{T1.value (↑) = T2.value (↑) / F.value (↑) }
-> T2 % F	{T1.value (↑) = T2.value (↑) % F.value (↑) }
-> F	{T1.value (↑) = F.value (↑) }
F -> (E)	{F.value(↑)= E.value (↑)}
->number	{F.value (↑) = number}

Here E,T,F have Synthesized attribute ‘value’. The Start symbol S will then have final value of expression evaluated.

Eg:

(1+5)/3-2*4





```
aneri@DESKTOP-64GOF59:/mnt/d/semester7/cd_lab/cd_lab8$ ./a.out
(1+5)/3-2*4

Evaluated Value=-6
aneri@DESKTOP-64GOF59:/mnt/d/semester7/cd_lab/cd_lab8$
```