CS 302 - HW4 - Elif Cemre Durgut - 26493

1.

a) E : expression
T : term
P : parenthesis
I : integers
. : decimal means

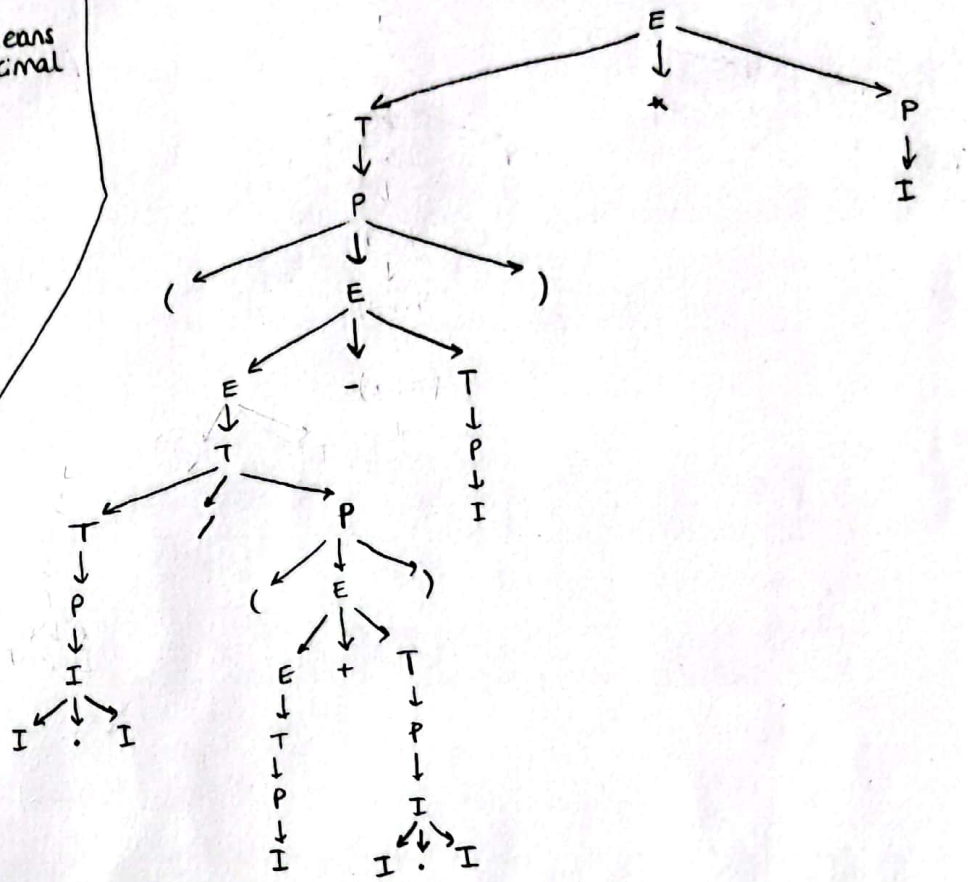
$E \rightarrow E+T \mid E-T \mid T$

$T \rightarrow T*P \mid T/P \mid P$

$P \rightarrow (E) \mid I$

$I \rightarrow I.I \mid I$

b) $(0.23 / (5+3.1) - 20)^{*} 2$



2. Priority order:
1) Parantheses
2) NOT
3) AND, OR

E : expression

$E \rightarrow E \wedge T \mid E \vee T \mid T$

$T \rightarrow \neg P \mid P$

$P \rightarrow (E) \mid V$

$V \rightarrow x \mid y \mid z \mid T \mid F$

5.1.2

b) Leftmost:

$S \rightarrow A1B \rightarrow 1B \rightarrow 10B \rightarrow 100B \rightarrow 1001B \rightarrow 1001$

Rightmost:

$S \rightarrow A1B \rightarrow A10B \rightarrow A100B \rightarrow A1001B \rightarrow A1001 \rightarrow 1001$

c) Leftmost:

$S \rightarrow A1B \rightarrow 0A1B \rightarrow 00A1B \rightarrow 000A1B \rightarrow 0001B \rightarrow 00011B \rightarrow 00011$

Rightmost:

$S \rightarrow A1B \rightarrow A11B \rightarrow A11 \rightarrow 0A11 \rightarrow 00A11 \rightarrow 000A11 \rightarrow 00011$

5.1.7

a) Proposition:

P(n): no string in L(G) of length n has ba as a substring.

Base case: P(1) L(G) of length 1 could be either b or a which, clearly do not contain ba.

Assuming that L(G) of length n has no ba, we need to prove it for string w of length n+1

The first derivation should be either aS or bS.

for aS:

S is replaced by some string w' in the next derivations.

$w = aS \quad |w| = n+1$

$\downarrow$

$w = aw' \longrightarrow$ so $|w'| = n \rightarrow$ we know that there is no ba in a string of length n. //

There cannot be string which contains ba.

b) It is defined as some number of a's followed by some number of b's. $\Rightarrow a^n b^m$ where $n, m \geq 0$

5.1.3 In regular languages, the production forms are:

$A \rightarrow t$ where t is a terminal

$A \rightarrow tB$ where t is " " and B is nonterminal

$A \rightarrow \epsilon$ where ϵ is empty string

} A CFG accepts all these three types of production rules.

5.1.4

a) To do so, I'll show an ϵ-NFA that simulates rightmost derivations. Suppose that we are now at here:

$ab\ldots cD \Rightarrow ab\ldots cdE$ which uses $D \Rightarrow dE$.

The corresponding DFA can imitate this step by going from state D to state E with the symbol d.

b) In regular languages, the automata starts to process string by starting from the leftmost bit and traverses it towards right.

$A \rightarrow aB \rightarrow abC \rightarrow abcD \rightarrow abcdE \rightarrow \ldots$