CSC 344 BNF Assignment

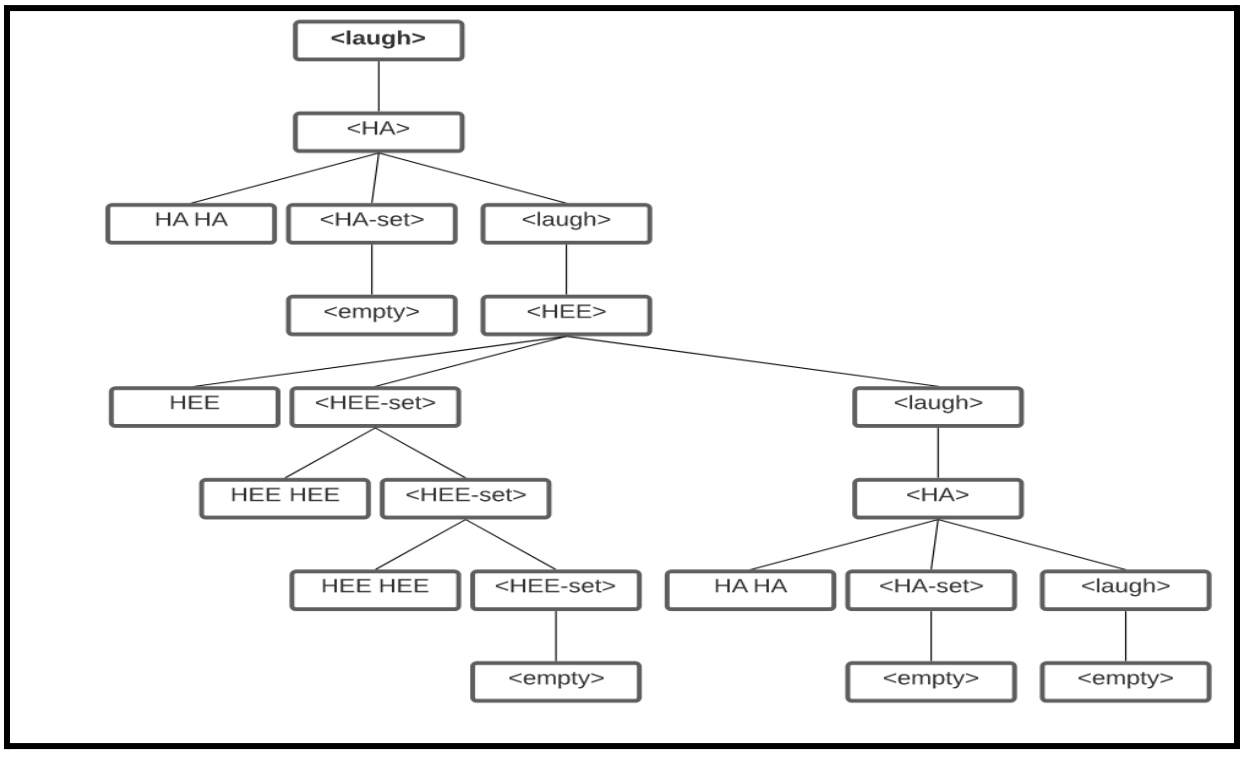
Abstract: This assignment is to build the BNF grammar and to draw some parse trees. The definition of the BNF grammar differs from person to person and helps to come up with some ideas to build our own definition of BNF grammar.

Problem 1: Laughter

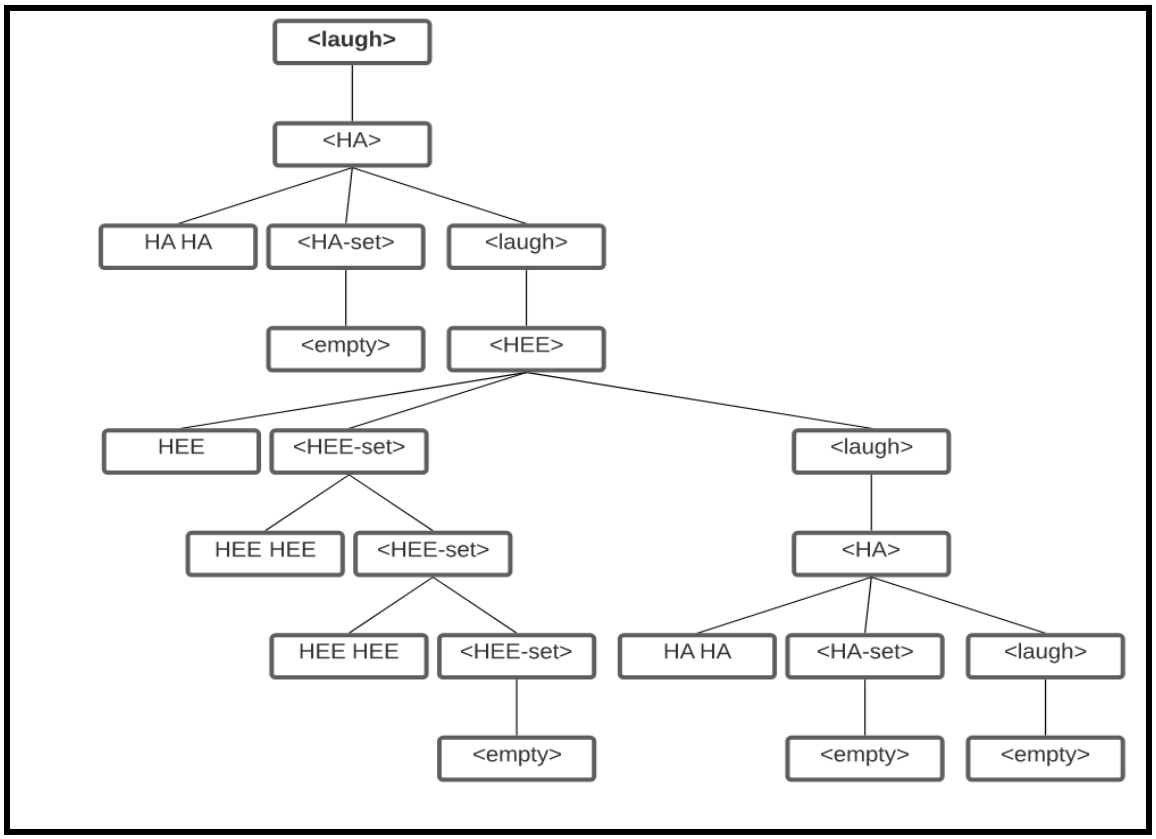
1. Write a BNF grammar for this language. Constraint: Refine all nonterminal symbols.

<laugh> : : = <HA> | <HEE> | <empty>
<HA> : : = HA HA <HA-set> <laugh> | <empty>
<HEE> : : = HEE <HEE-set><laugh> | <empty>
<HA-set> : : = HA HA <HA-set> | <empty>
<HEE-set> : : = HEE HEE <HEE-set> | <empty>

2. Draw a parse tree for the following sentence: HA HA HEE HEE HEE HEE HEE HA HA



3. Draw a parse tree for the following sentence: HEE HA HA HA HA HA HA

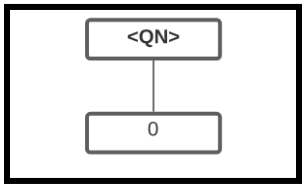


Problem 2: SQN (Special Quaternary Numbers)

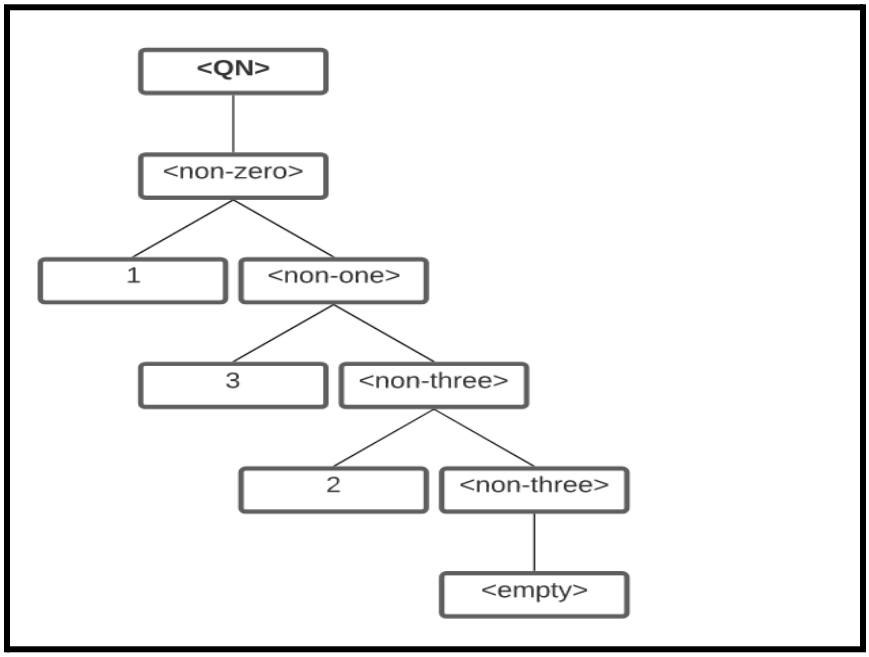
1. Write a BNF grammar description of the SQN language.

<QN> : : = 0 | <non-zero> | <empty>
<non-zero> : : = 1 <non-one> | 2 <non-two>| 3 <non-three> | <empty>
<non-one> : : = 0 <non-zero>|2<non-two>|3 <non-three> | <empty>
<non-two> : : = 0 <non-zero | 1 <non-one> | 3 <non-three> | <empty>
<non-three> : : = 0 <non-zero>| 1 <non-one> | 2<non-two> | <empty>

2. Draw a parse tree, consistent with the BNF grammar that you crafted, for the following sentence: 0

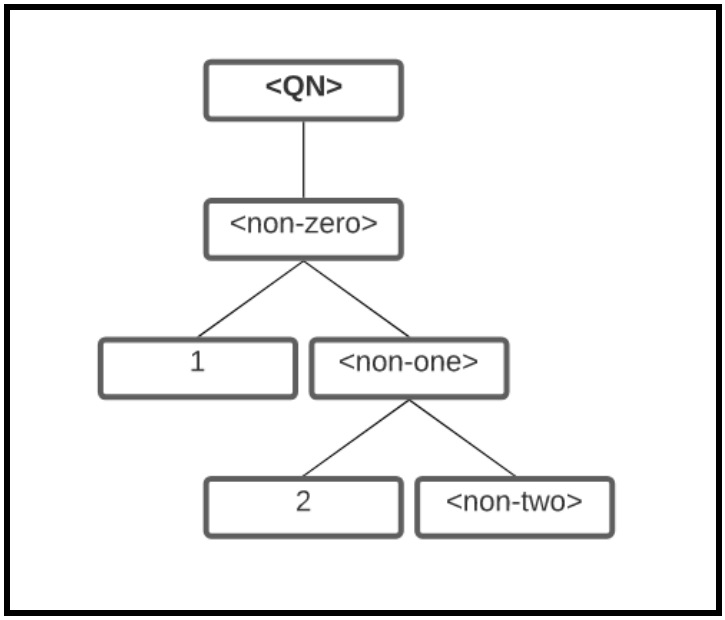


3. Draw a parse tree, consistent with the BNF grammar that you crafted, for the following sentence: 132



4. Explain, in precise terms, why you cannot draw a parse tree, consistent with the BNF grammar that you crafted, for the string: 1223

It's because for making the first number '2' we have 2 options i.e. 2 and <non-two> in which <non-two> contains other numbers but not 2. So it gets stuck at that point as shown in the tree diagram below:

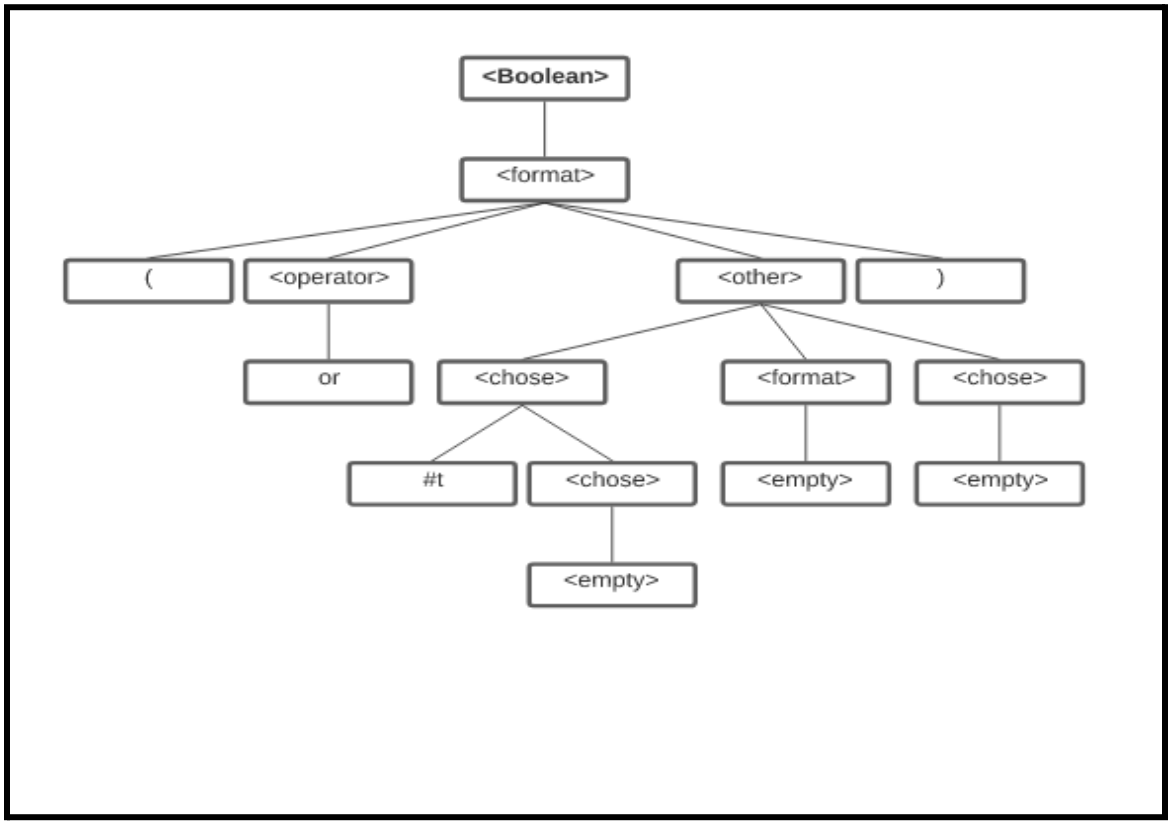


Problem 3: BXR

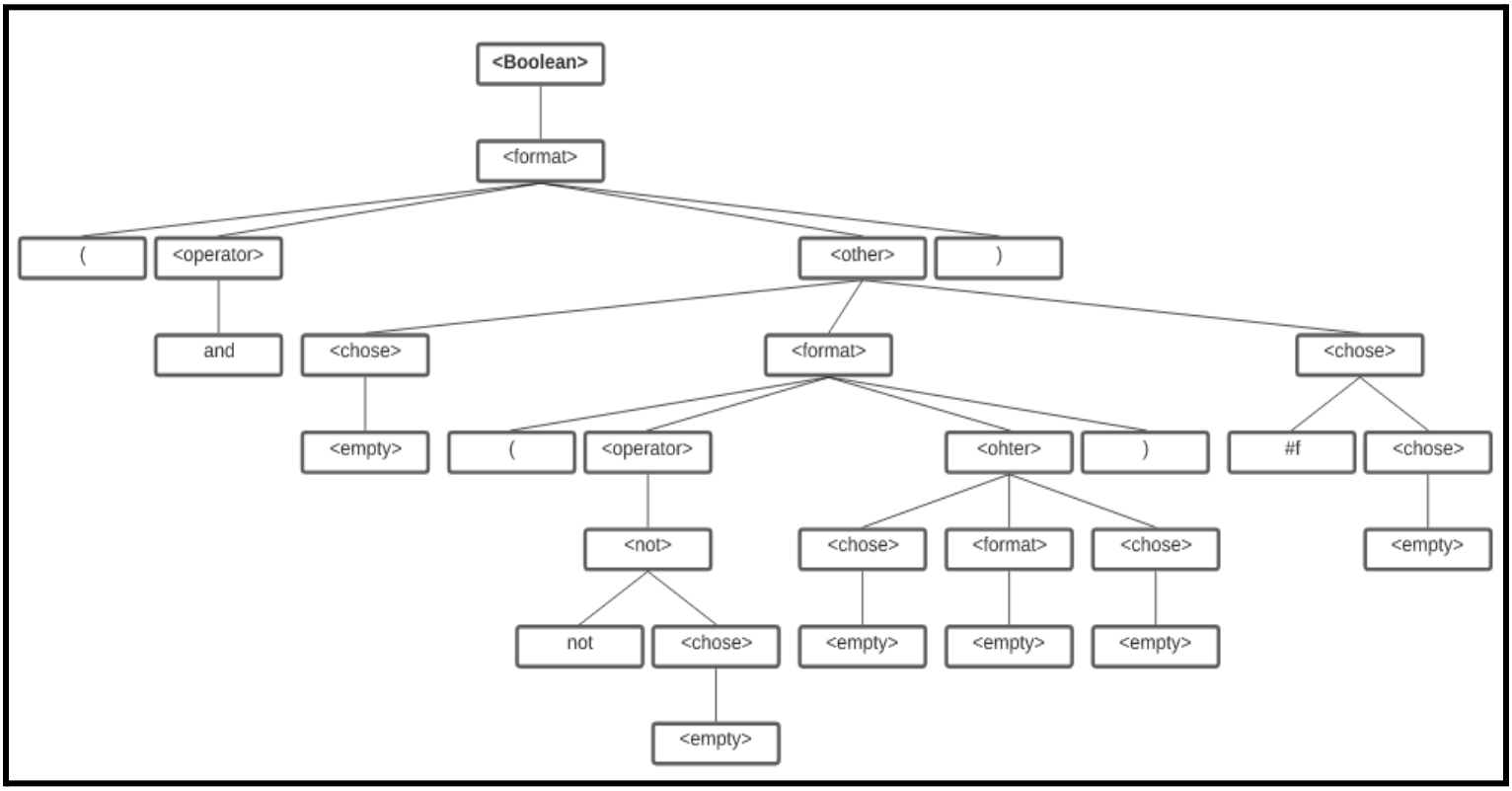
1. Write a BNF grammar description of this language.

<Boolean> : : = <chose> | <choose> | <format> | <empty>
<format> : : = (<operator> <other>) | <empty>
<chose> : : #t <chose> | #f <chose> | <empty>
<choose> : : = #t | #f
<operator> : : = <not> | or | and
<other> : : = <chose> <format><chose>
<not> : : = not<choose>

2. Draw a parse tree, consistent with the BNF grammar that you crafted, for the following sentence: (or #t)



3. Draw a parse tree, consistent with the BNF grammar that you crafted, for the following sentence:
(and (not #t) #f)

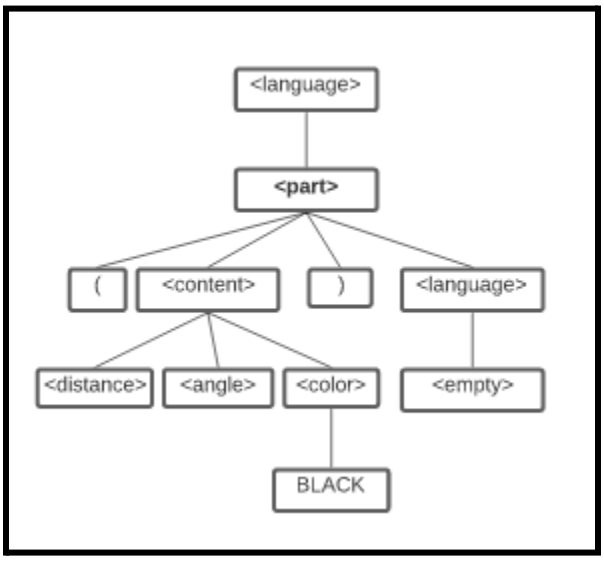


Problem 4 - LSS (Line Segment Sequences)

1. Write a BNF grammar for this language. Constraint: Refine all nonterminal symbols except the one corresponding to distance and the one corresponding to the angle.

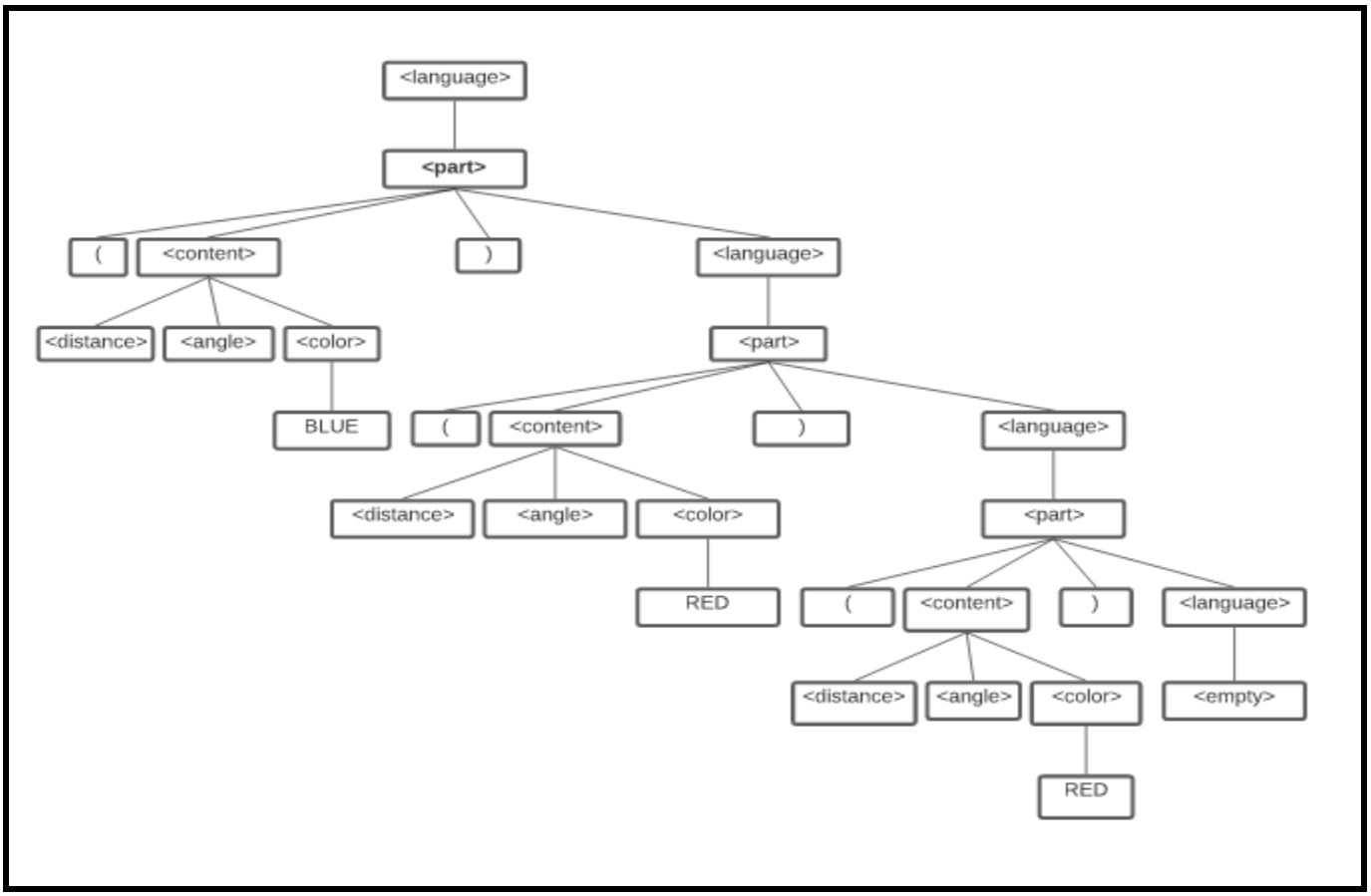
<language> : : = <part> | <empty>
<part> : : =(<content>) <language>
<content> : : = <distance> <angle><color>
<color> : : = RED | BLACK | BLUE

2. Draw a parse tree for the following sentence:
(120 95 BLACK)



3. Draw a parse tree for the following sentence:

(70 180 BLUE) (770 187 RED) (191 145 RED)



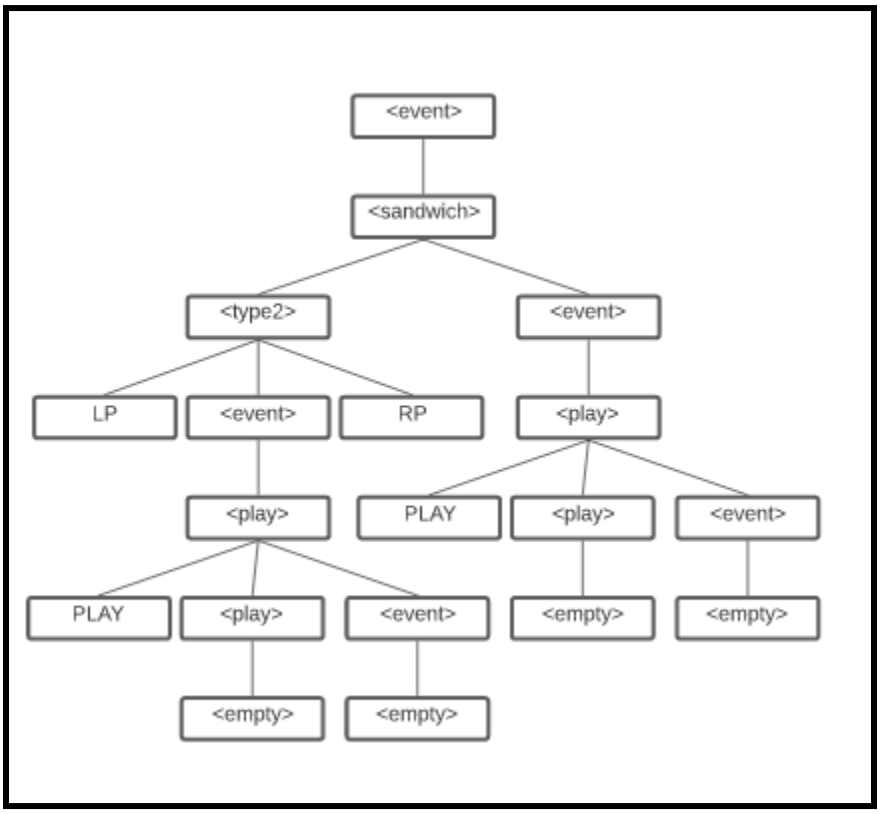
Problem 5 - M-Lines

1. Write a BNF grammar description of this language

<event> : : = <play> | <rest> | <sandwich> | <empty>
<play> : : = play <play><event> | <empty>
<rest> : : = rest <rest><event>4> | <empty>
<sandwich> : : = <type1><event> | <type2> <event> | <type3><event> | <type4><event> | <type5><event> | <type6><event>

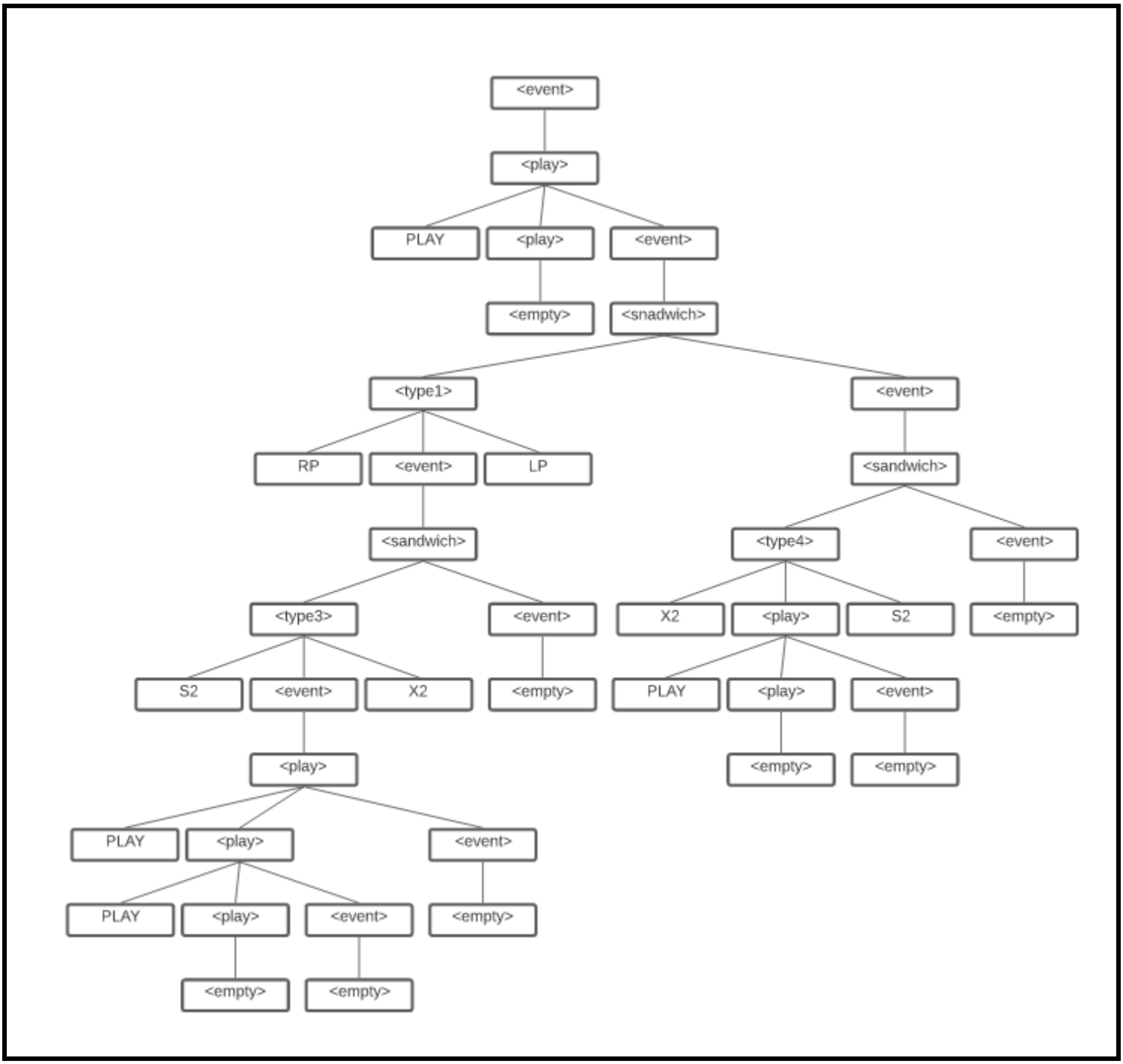
<type1> : : = RP <event> LP
<type2> : : = LP <event> RP
<type3> : : = S2 <event> X2
<type4> : : = X2 <event> S2
<type5> : : = S3 <event> X3
<type6> : : = X3 <event> S3

2. Draw a parse tree, consistent with the BNF grammar that you crafted, for the following sentence: LP PLAY RP PLAY



-

3. Draw a parse tree, consistent with the BNF grammar that you crafted, for the following sentence:
PLAY RP S2 PLAY PLAY X2 LP X2 PLAY S2



Problem 6 - BNF?

BNF stands for Backus Naur Form, which is a notation used to describe the grammar of a programming language. It defines the rules for constructing valid sentences or expressions in a language, similar to how grammar rules define how to construct sentences in a spoken language. It consists of a set of production rules that specify how valid programs can be constructed from basic elements, such as keywords, operators, and variables. BNF is used to develop the syntax of programming languages and to create compilers that can interpret or compile programs written in that language. It's an important concept for computer science students to understand, as it helps them better understand how programming languages work and how to write code that is both readable and understandable.