Declan ONUNKWO 02/14/2023

CSC344 – Programming Languages

Assignment: **Backus-Naur Form (BNF) Assignment.**

ABSTRACT

The aim of this assignment is to familiarize oneself with Backus-Naur Form and learn about its use in computer science. This assignment will cover the basics of BNF, including its syntax, symbols, and rules for defining a programming language. We also learn how BNF is used to describe the structure and grammar of a language, and how it can be used to generate a parser for that language. The first task has constraints which is helps one to better understand how to create a BNF with strict rules, while the rest of the tasks are quite straightforward. The idea is repetition is how we learn.

PROBLEM 1

The BNF Grammar

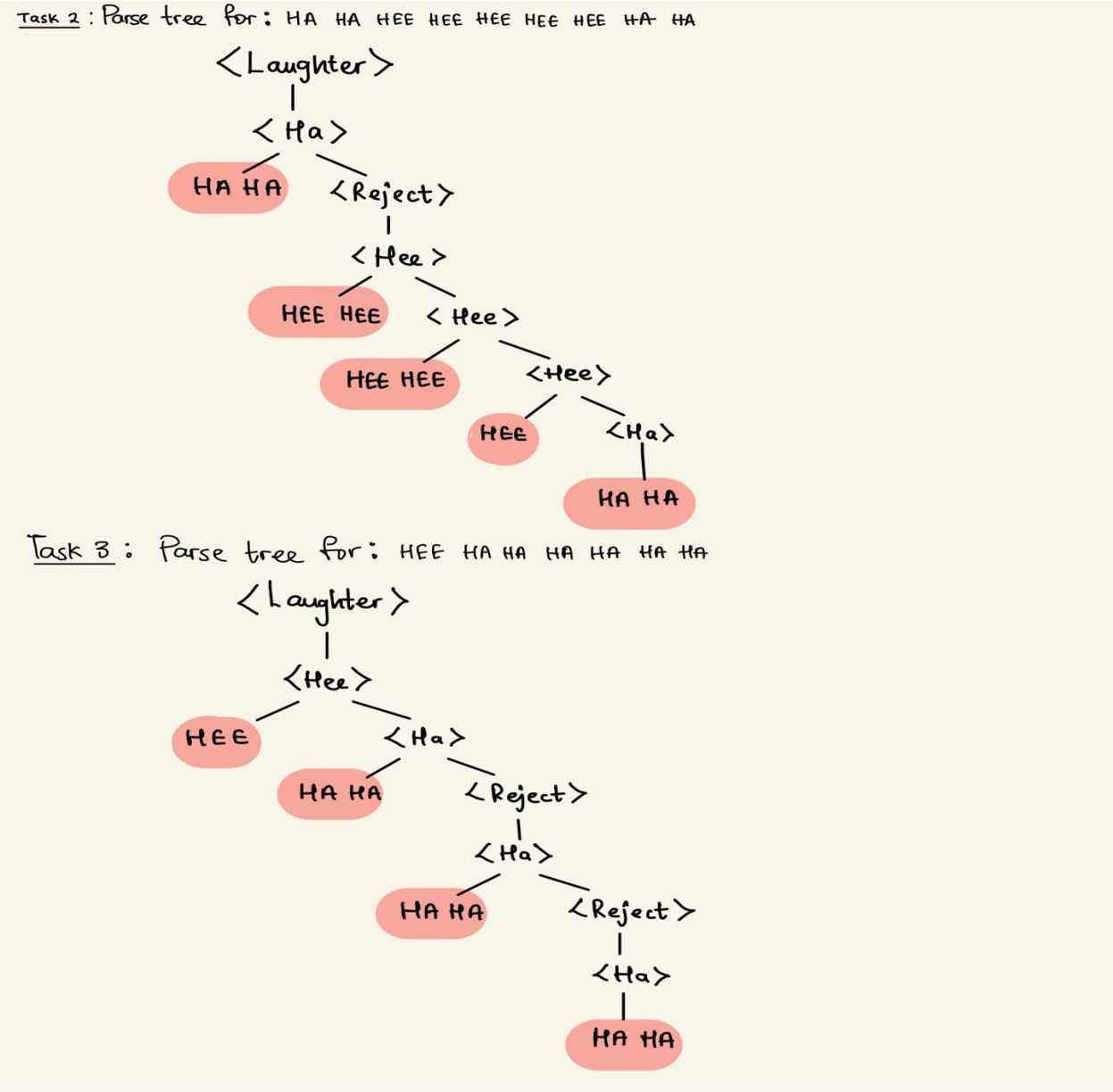
<Laughter> ::= <Ha> | <Hee> | <empty>

<Ha> ::= "HA HA " <Reject> | "HA HA "

<Hee> ::= "HEE " | "HEE " <Ha> | "HEE HEE " <Hee>

<Reject> ::= <Ha> | <Hee>

The Parse Trees



PROBLEM 2

The BNF Grammar

<SQN> ::= "0" | <nonZero>

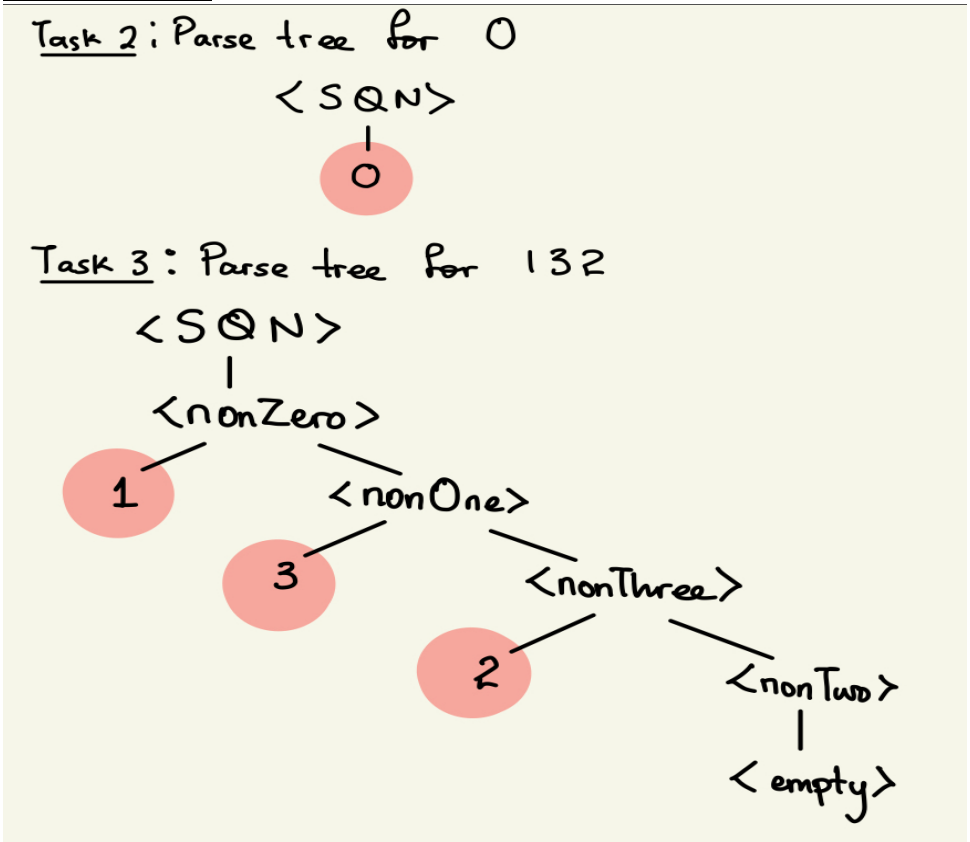
<nonZero> ::= "1" <nonOne> | "2" <nonTwo> | "3" <nonThree> | <empty>

<nonOne> ::= "0" <nonZero> | "2" <nonTwo> | "3" <nonThree> | <empty>

<nonTwo> ::= "0" <nonZero> | "1" <nonOne> | "3" <nonThree> | <empty>

<nonThree> ::= "0" <nonZero> | "1" <nonOne> | "2" <nonTwo> | <empty>

The Parse Trees



Task 4: While attempting to create a consecutive “2” with the BNF language above, it was not possible because the production does not permit it.

The production <nonTwo> ::= "0" <nonZero> | "1" <nonOne> | "3" <nonThree> | <empty>

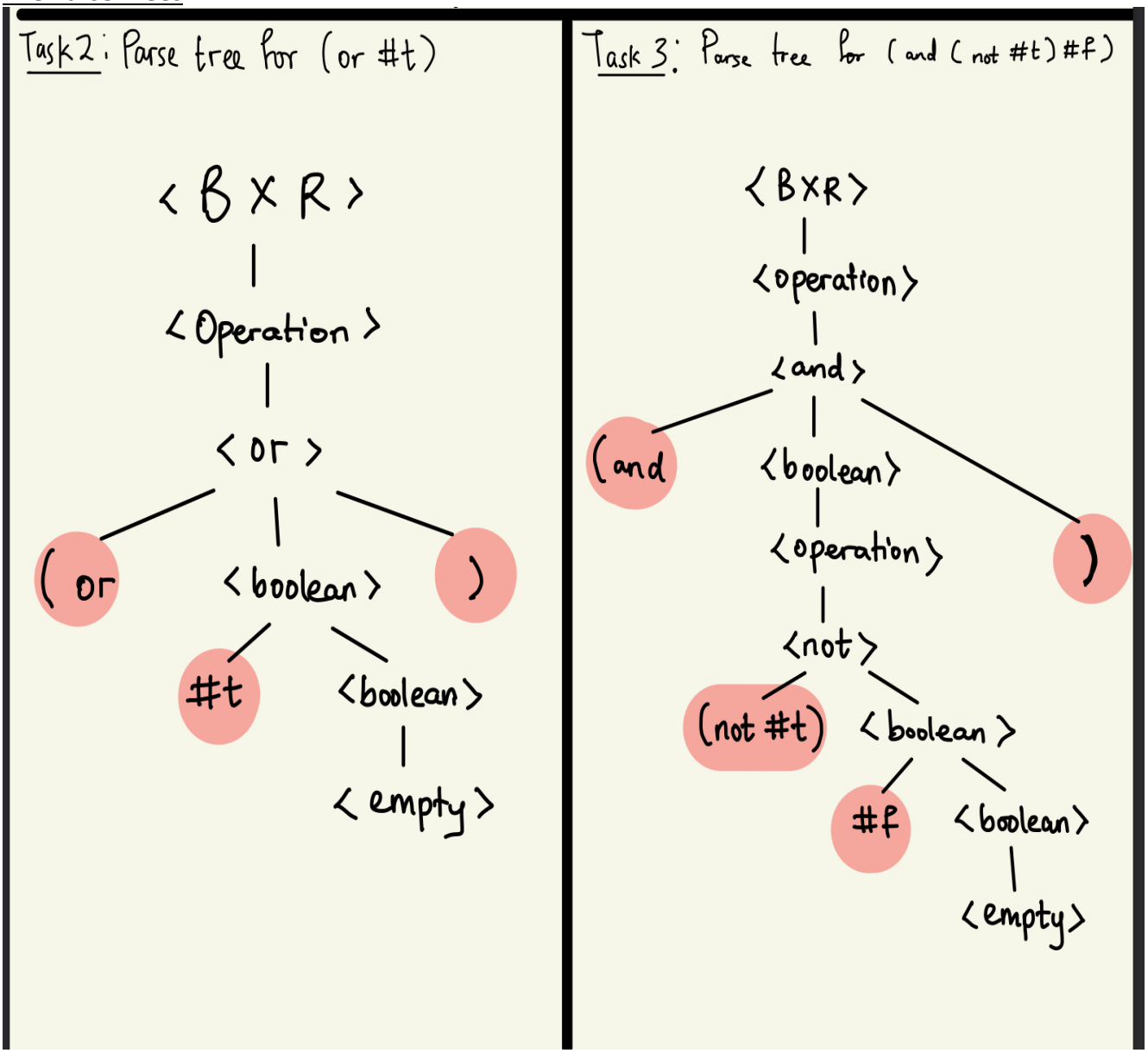
The token “2” is not present in the above.

PROBLEM 3

The BNF Grammar

<BXR> ::= <operation> | "#t" | "#f"
<operation> ::= <and> | <or> | <not>
<and> ::= "(and "<boolean> ")" | "(and)"
<or> ::= "(or "<boolean> ")"| "(or)"
<not> ::= "(not #t)" | "(not #f)" | "(not #t)"<boolean> | "(not #f)"<boolean>
<boolean> ::= "#t "<boolean> | "#f "<boolean> | <operation> | <empty>

The Parse Trees



PROBLEM 4

The BNF Grammar

<LSS> ::= <sequence> | <empty>

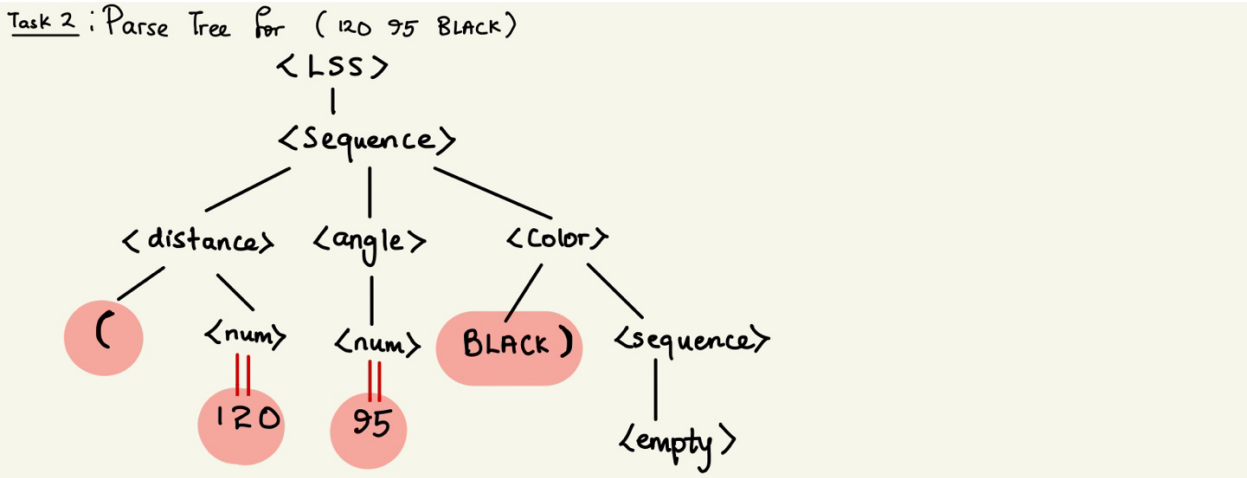
<sequence> ::= <distance> <angle> <color> | <empty>

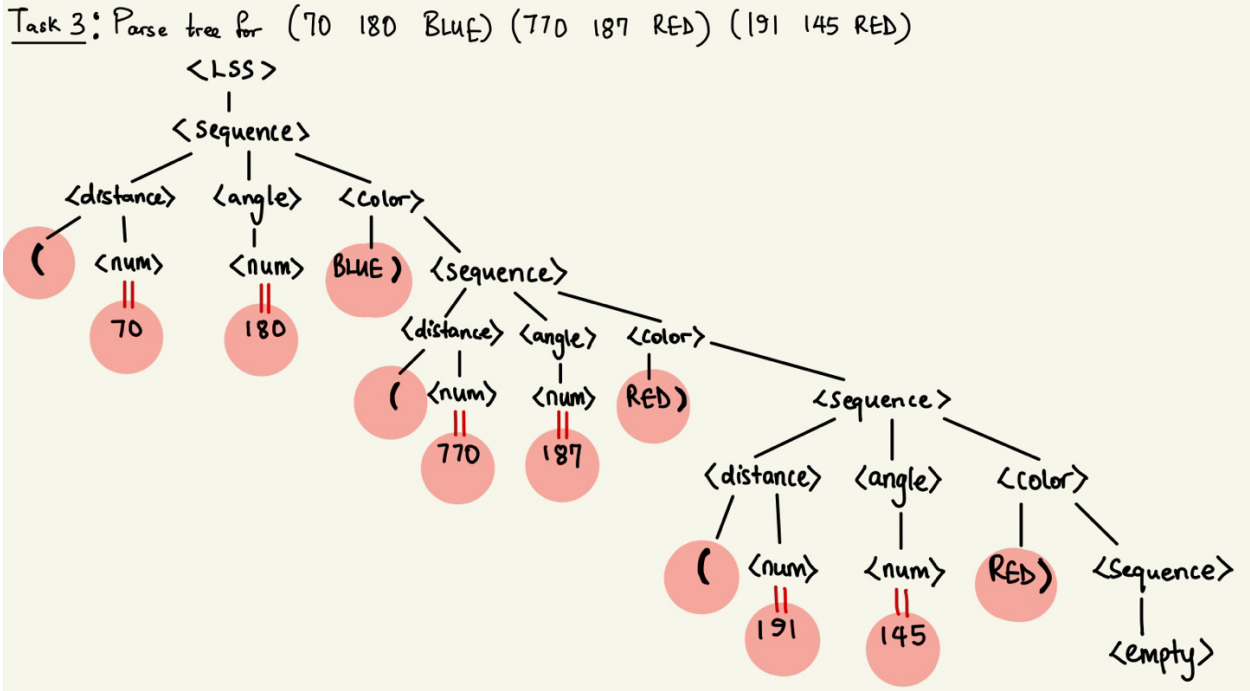
<distance> ::= "(" <num>

<angle> ::= <num>

<color> ::= " RED) " <sequence> | " BLACK) " <sequence> | " BLUE) " <sequence>

The Parse Trees





PROBLEM 5

The BNF Grammar

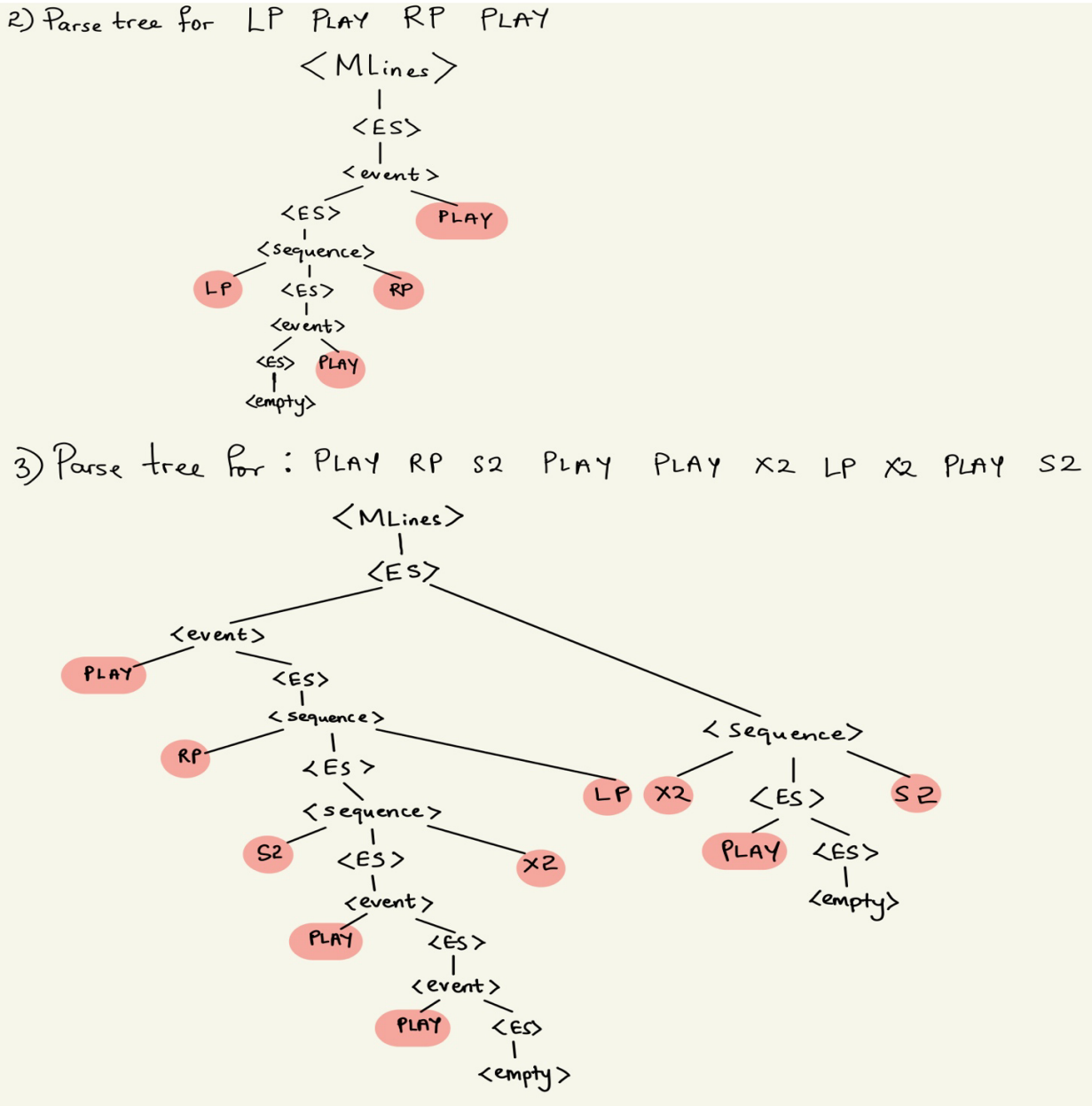
<MLines> ::= <ES> | <empty>

<ES> ::= <event> | <sequence> | <event><sequence> | <sequence><event> | <empty>

<event> ::= "PLAY " <ES> | "REST " <ES> | <ES> "PLAY " | <ES> "REST " | <empty>

<sequence> ::= "RP " <ES> "LP " | "LP " <ES> "RP " | "S2 " <ES> "X2 " | "X2 " <ES> "S2 " | "S3 " <ES> "X3 " | "X3 " <ES> "S3 " | <empty>

The Parse Trees



PROBLEM 6

BNF stands for Backus-Naur Form. It is a notation used in computer science to formally describe the syntax of programming languages. It specifies the structure and rules of a language, making it easier for programmers to write code that is both accurate and readable. BNF is important because it provides a way to define the grammar of a language, ensuring consistency and reducing the risk of errors. If the rules of the grammar are followed, every possible permutation is achievable. BNF is also used in the development of many programming languages and compilers.