BNF Assignment:

Cameron Francois
2/14/2023
CSC 344

Abstract:
This assignment is all about BNF. You will be asked to compose some BNF grammars for given languages. You will be ask to draw some BNF parse trees. You will be asked to describe BNF in English, in a straightforward, compelling manner.

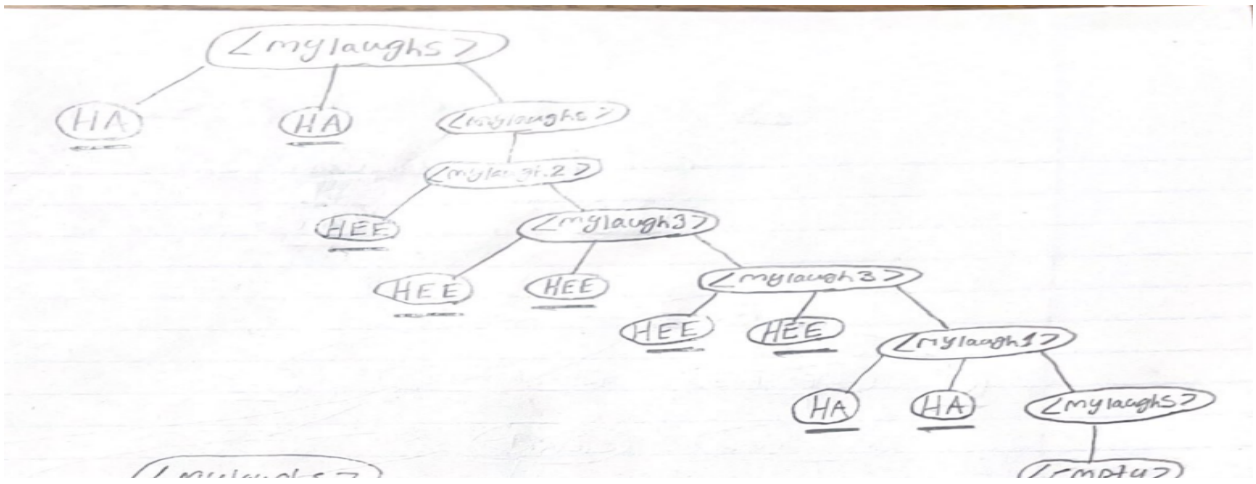
Problem 1 - Laughter:

Start = {mylaughs}
Tokens = {HA, HEE}
Nonterminals = {mylaughs, mylaugh1, mylaugh2, mylaugh3, empty}

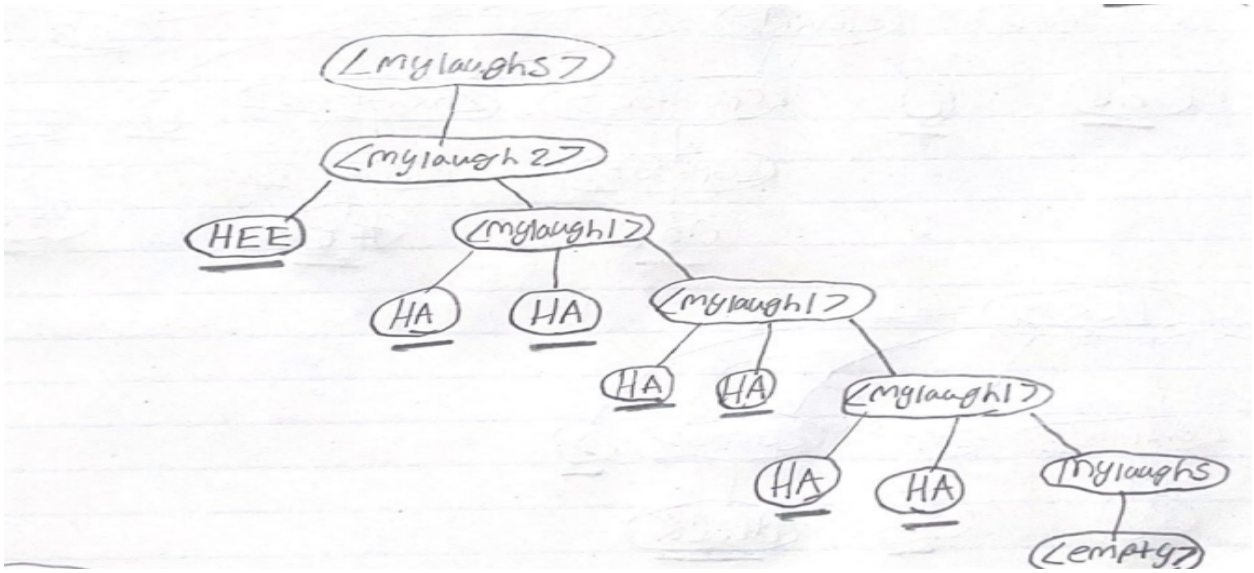
Productions:

<mylaughs> ::= <mylaugh1> | <mylaugh2> | <mylaugh3> | <empty>
<mylaugh1> ::= HA HA | HA HA <mylaugh1> | HA HA <mylaughs>
<mylaugh2> ::= HEE | HEE <mylaugh1> | HEE <mylaugh3>
<mylaugh3> ::= HEE | HEE HEE <mylaugh3> | HEE HEE <mylaugh1> | <mylaughs>

Sentence 1: HA HA HEE HEE HEE HEE HEE HA HA



Sentence 2: HEE HA HA HA HA HA HA



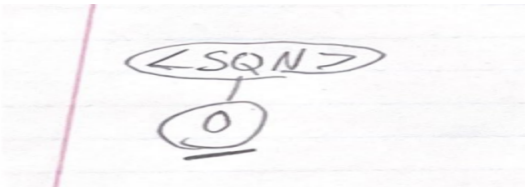
<u>Problem 2 - SQN</u>:

Start = {SQN}
Tokens = {0,1,2,3}
Nonterminals = {SQN, nz-sequnces, nz-seq0 nz-seq1, nz-seq2, nz-seq3, empty}

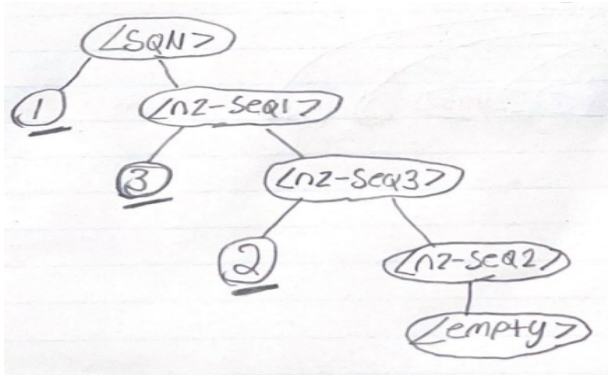
Productions:

<SQN> ::= 0 | <nz-sequences>
<nz-sequences> ::= 1 | 2 | 3 | 1 <nz-seq1> | 2 <nz-seq2> | 3 <nz-seq3> |
<nz-seq1> ::= 0 <nz-seq0> | 2 <nz-seq2> | 3 <nz-seq3> | <empty>
<nz-seq2> ::= 0 <nz-seq0> | 1 <nz-seq1> | 3 <nz-seq3> | <empty>
<nz-seq3> ::= 0 <nz-seq0> | 1 <nz-seq1> | 2 <nz-seq2> | <empty>
<nz-seq0> ::= <nz-sequences> | <empty>

Sentence 1: 0



Sentence 2: 132



Sentence 3: 1223
This example does not work because we set up our grammar to avoid having repeated digits in our sentences hence why this can't happen due to '2' '2' happening back to back.

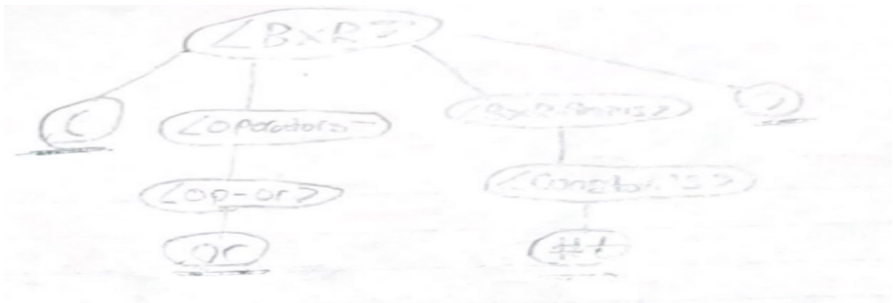
Problem 3 - BXR:

Start = { BXR }
Tokens = { (,), #t, #f, and, or, not }
Nonterminals = {BXR, BXR-terms, constants, operators, op-and, op-or, op-not, empty }

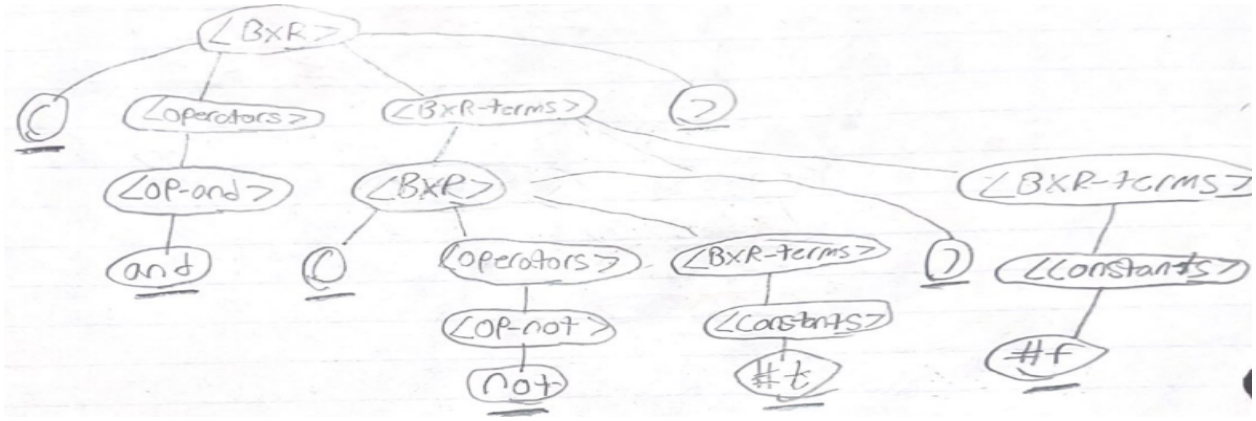
Productions:

<BXR> ::= <constants> | (<operators> <BXR-terms>)
<BXR-terms> ::= <BXR-terms> <BXR> | constants | <empty>
<constants> ::= #t | #f
<operators> ::= <op-and> | <op-or> | <op-not>
<op-and> ::= and
<op-or> ::= or
<op-not> ::= not

Sentence 1: (or #t)



Sentence 2: (and (not #t) #f)



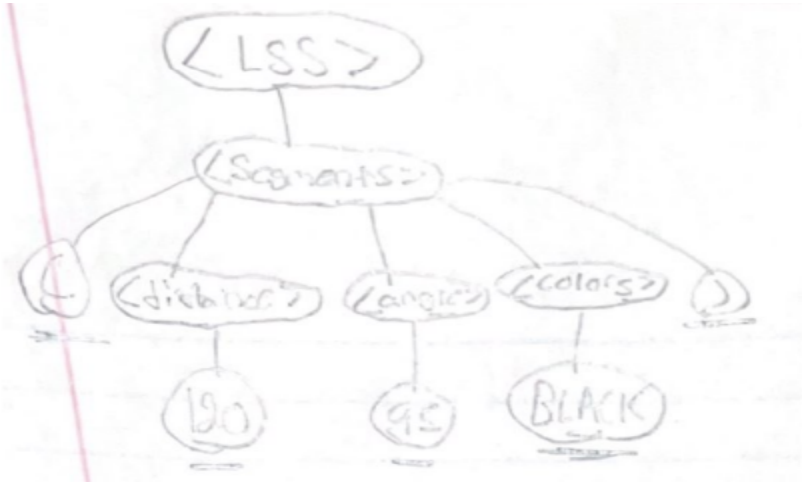
<u>Problem 4 - LSS(Line Segment Sequences)</u>:

Start = { LSS }
Tokens = { (,), RED, BLUE, BLACK }
Nonterminals = { segments, distance, angle, colors, empty }

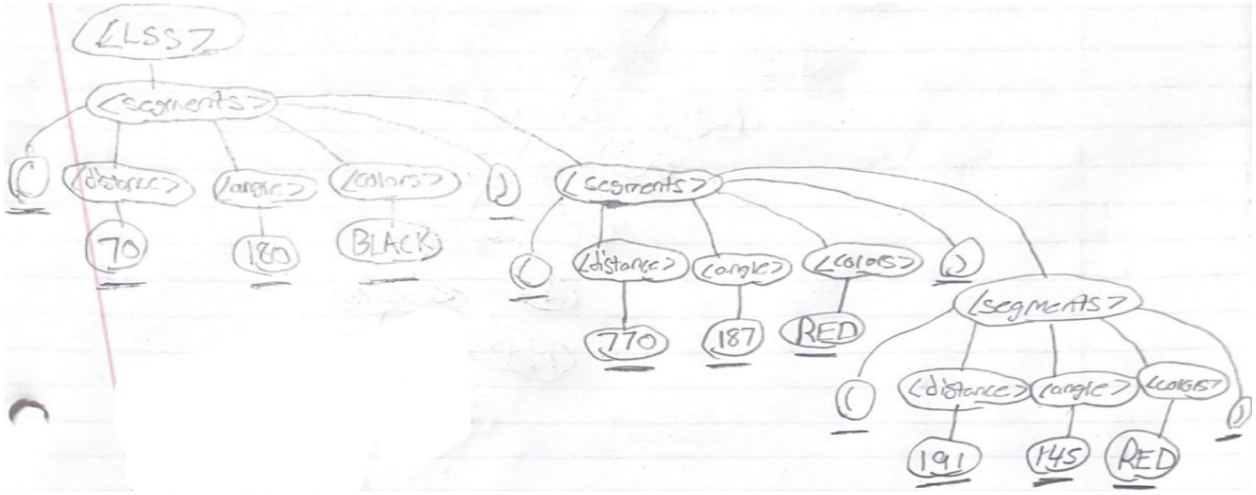
Productions:

<LSS> ::= <segments>
<segments> ::= (<distance> <angle> <colors>) <segments> | (<distance> <angle> <colors>)
<colors> ::= RED | BLUE | BLACK

Sentence 1: (120 95 BLACK)



Sentence 2: (70 180 BLACK) (770 187 RED) (191 145 RED)



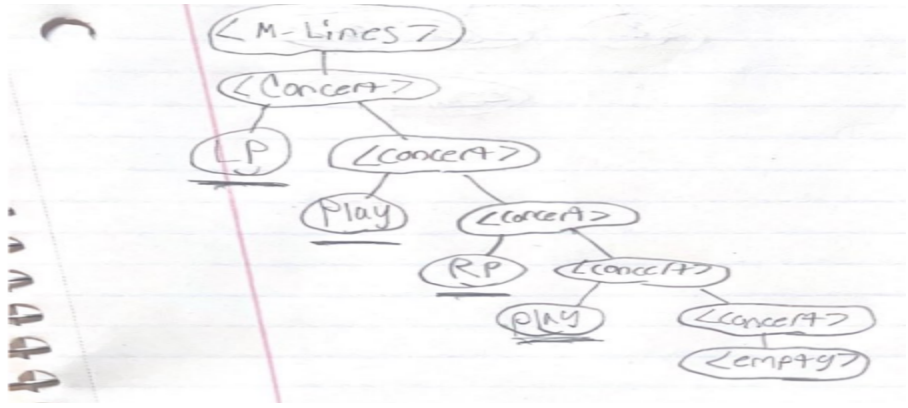
<u>Problem 5 - M-Lines</u>:

Sart = { M-Lines }
Tokens = { RP, LP, S2, S3, X2, X3, PLAY, REST }
Nonterminals = { Concert, empty }

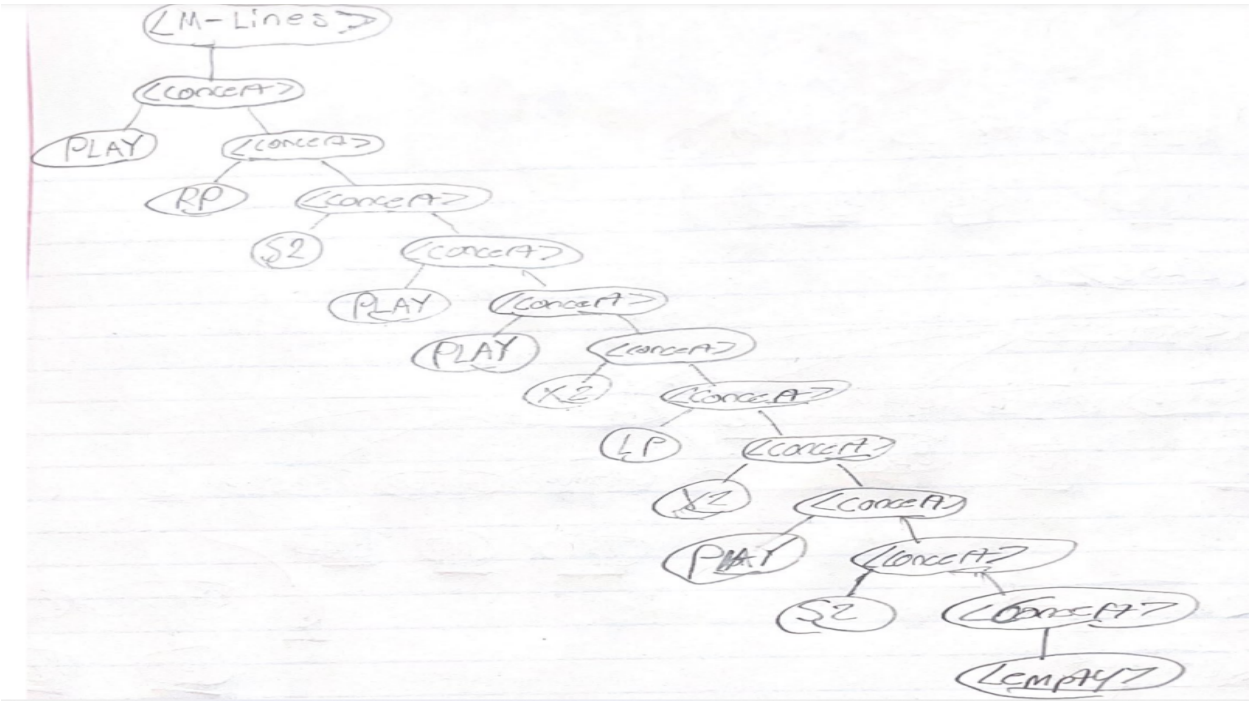
Productions:

<M-Lines> ::= <Concert>
<Concert> ::= RP <Concert> | LP <Concert> | S2 <Concert> |
S3 <Concert> | X2 <Concert> | X3 <Concert> | PLAY <Concert> | REST
<Concert> | <empty>

Sentence 1: LP PLAY RP PLAY



Sentence 2: PLAY RP S2 PLAY PLAY X2 LP X2 PLAY S2



<u>Problem 6 - BNF?</u>:

What is BNF? Well, first off it stands for Backus-Naur Form, which is a form of grammar that is a way of helping programmers to label certain syntax rules of a language. BNF allows developers to use symbols to express their current vocabulary in their language. Some of the key components of BNF are the tokens, nonterminals. The nonterminals is where the developers can define, and tokens are already part of the given language. Other important components are the start symbol which is what represents for the given language and productions are used to connect all the components together.