Jason Lu.
912237602.

1.

A ::= [B,A] | B.

B ::= C | (A;C).

C ::= {C} | D.

D ::= a | b | c.

a). [c, (b;a)]

A → [B,A].
→ [C , B].
→ [D, (A;C)].
→ [c, (B;D)].
→ [c, (D;D)].
→ [c, (b;a)].

A can generate.

b). ([(a;b) , c]; a)

A → B
→ (A;C).
→ ([B,A]; C).
→ ([(A;C),B]; C)
→ ([(B;D), C]; D).
→ ([(D;D), D]; D).
→ ([(a;b), c]; a).

A, B can generate.

c). [(a;c),[[b;[a,c]],b]]

A → [B,A].
→ [(A;C),[B,A]]
= [(B;C),[(A;C), A]].

since there is no way for C to derive into [a, c]. Thus, there is no way we can derive this based on given grammar [none].

2.

a).

<integer> ::= [<sign>] <unsigned>

<unsigned> ::= {<digits>} <digits>

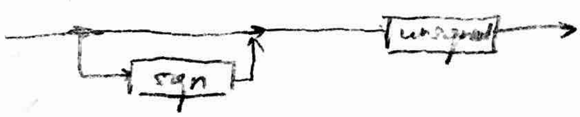
<digits> ::= {<digit>} <digit>

<digit> ::= 0 | 1 | ... | 9.

<sign> ::= + | -

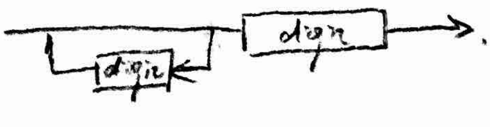
b)

①

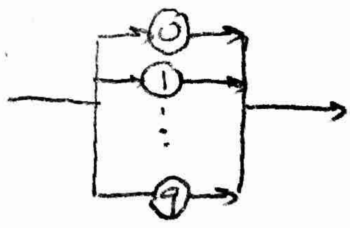


②

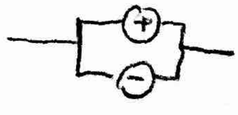
③



④



⑤



3. what language is generated by each of BNF grammars below.

a).

<S> ::= 0 <S> 1111 | empty

$L = \{ 0^k (1)^{4k} \mid k \geq 0 \}$.

b)

<S> ::= <x> | <y> | empty

<x> ::= 1 <x> 00 | empty

<y> ::= 0 <y> 1 | empty

$L_x = \{ 1^k 0^{2k} \mid k \geq 0 \}$.

$L_y = \{ 0^k 1^k \mid k \geq 0 \}$.

$L = \{ L_x \cup L_y \}$

c)

<S> ::= <x> | <y>

<x> ::= 0 <x> 1 | <x1>

<x1> ::= 0 <x1> | 0

<y> ::= 0 <y> 11 | <y1>

<y1> ::= <y1> 1 | 1.

$L_{x1} = \{ 0^k \mid k \geq 1 \}$.

$L_{y1} = \{ 1^k \mid k \geq 1 \}$.

$L_x = \{ 0^x 1^x \text{ or } 0^k \mid k \geq 1 \ \& \ x \geq 1 \}$

$L_y = \{ 0^x 1^{2x} \text{ or } 1^k \mid k \geq 1 \ \& \ x \geq 1 \}$.

$L = \{ L_x \cup L_y \}$

4.

Given the following grammar.

<stmt> ::= if <expr> then <stmt>

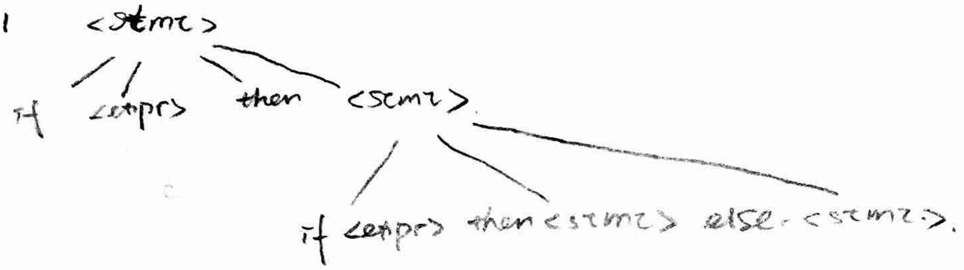
| if <expr> then <stmt> else <stmt>.

| other

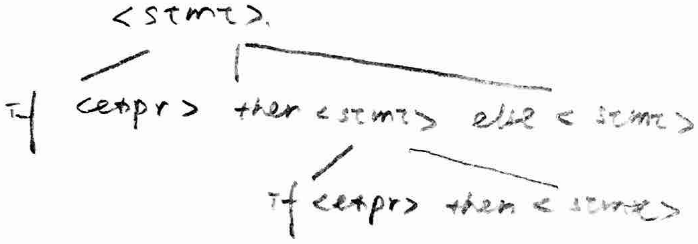
<expr> ::= true | false

a).

Tree 1



Tree 2



string: if <expr> then if <expr> then <stmt> else <stmt>

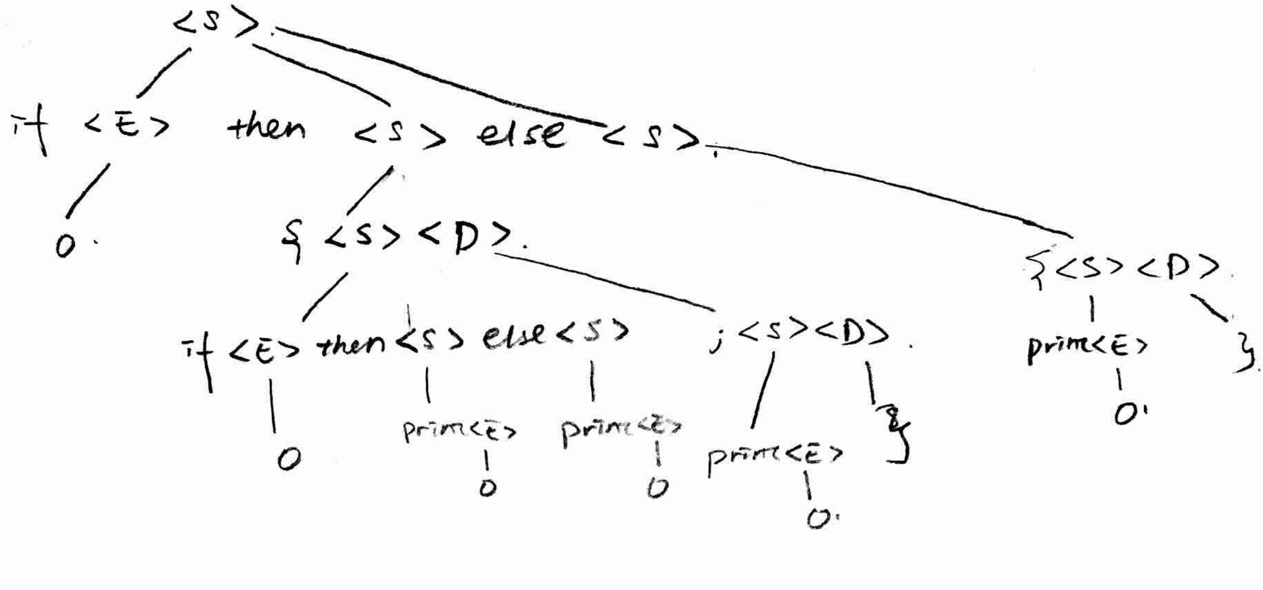
b).

<stmt> ::= if <expr> then <NT> | <NT> | other.

<NT> ::= if <expr> then <NT> else <stmt> | other.

<expr> ::= true | false.

5.



6.

a. Alogrithm:

Base Case: if the nonterminal can be derive into any terminal. then remove the nonterminal symbol from the nonterminal array. Return the nonterminal array.

Recursion: for (each nonterminal in the nonterminal array) {
  if (it can be derive into terminal) {
    Remove itself from the nonterminal array
  } else if (it can only be derive into nonterminals (doesn't contain itself)
    pass that nonterminals into the function again
  } else if (it can only be derive into nonterminals that contain itself) {
    keep reading & don't recurse.
}

b. Read the first line "<S> ::= 0 | <A> | <C>" since <S> can be derive into terminal 0. Remove <S> from nonterminal array.

The second line "<A> ::= <A><B>". Since <A> can only be derive into nonterminals <A><B>, which contains <A> itself, so it stays in the nonterminal array and we move on.

After we read the 3rd and 4th line, we see that both <B> and <C> can be removed from nonterminal array.

Finally we return the array nonterminal, which now only contains the useless nonterminal <A>