CMPSC 461 Spring 2022
Homework 1 Solution. Gang Tan. Please do not upload the document to an online site such as coursehero.com.

1. **(a)**

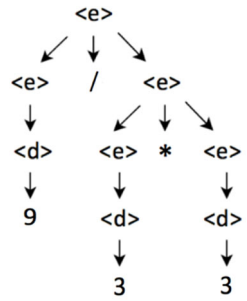
<e> **->** <e> / <e> **->** <d> / <e> **->** 9 / <e> **->** 9 / <e> * <e> **->** 9 / <d> * <e> **->**
9 / 3 * <e> **->** 9 / 3 * <d> **->** 9 / 3 * 3
Another possible answer is: <e> -> <e> * <e> -> …

1. **(b)**

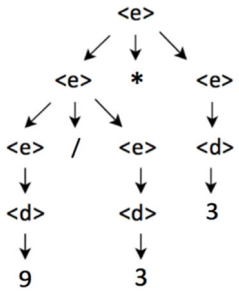
<e> **->** <e> * <e> **->** <e> * <d> **->** <e> * 3 **->** <e> / <e> * 3 **->** <e> / <d> * 3 **->**
<e> / 3 * 3 **->** <d> / 3 * 3 **->** 9 / 3 * 3
Another possible answer is: <e> -> <e> / <e> -> …

1. **(c)**

For (a):



For (b):

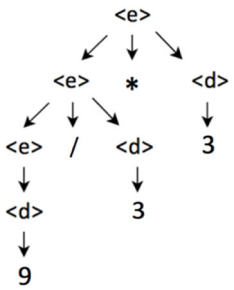


Note: depending on the answers for (b) and (c), it's possible that the parse trees for the two derivations are the same (one of the above).

1. **(d)**

<e> -> <d> | <e>/<d> | <e>*<d>
<d> -> 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

The parse tree for "9 / 3 * 3" based on the new grammar:



Argument for unambiguity: there are two choices to begin parsing 9/3*3:

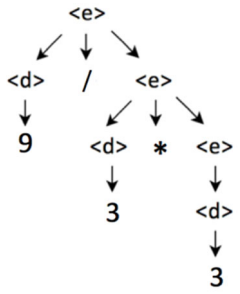
(1) <e> -> <e>/<d> or (2) <e> -> <e>*<d>

For option (1), it is impossible to get the expression 9/3*3 from <e> -> <e>/<d> -> <e>/3 -> <e>*<d>/3.
For option (2), it is shown above. Hence, (2) is the only choice.

1. (e)

<e> -> <d> | <d>/<e> | <d>*<e>
<d> -> 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

The parse tree for “9 / 3 * 3” based on the new grammar:

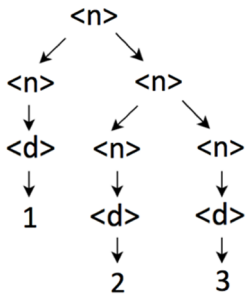


2.

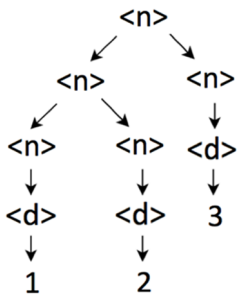
The grammar is ambiguous.

Example input: 123

Corresponding parse tree 1:



parse tree 2:



An equivalent unambiguous grammar:

<n> -> <n><d> | <d>
<d> -> 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

3. <S> -> ()) | (<S>))

4.

<e> -> <u> | <e> + <u> | <e> - <u>
<u> -> <d> | ~ <u>
<d> -> 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

5.

<email> -> <account> @ <subDomains> . <topDomain>
<account> -> <letter> | <account><letter> | <account><digit>
<subDomains> -> <letterDigitSeq> | <letterDigitSeq> . <subDomains>
<letterDigitSeq> -> <letter> | <digit>
| <letter><letterDigitSeq>
| <digit><letterDigitSeq>
<topDomain> -> com | org | gov | net

<letter> -> a | b | c | ... | z | A | B | C | ... | Z
<digit> -> 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

6. **BNF grammar:**

<SNFloat> -> <Float> | <Float>E<Exponent>
<Float> -> <NonZeroDigit> | <NonZeroDigit>.<Num>
<Exponent> -> <Num> | +<Num> | -<Num>
<Num> -> <Digit> | <Digit><Num>
<Digit> -> 0 | <NonZeroDigit>
<NonZeroDigit> -> 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9