CMPSC 461 Spring 2018
Homework 1 Solution. Gang Tan.

1. (a) leftmost derivation for the expression “2 * 3 * 6”

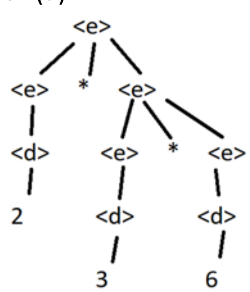
<e> -> <e>*<e> -> <d>*<e> -> 2*<e> -> 2*<e>*<e> -> 2*<d>*<e> ->
2*3*<e> -> 2*3*<d> -> 2*3*6

1. (b) rightmost derivation for “2*3*6”

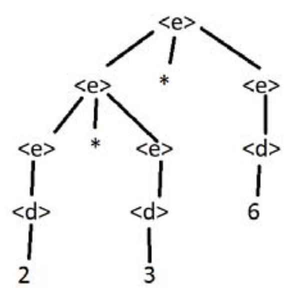
<e> -> <e>*<e> -> <e>*<d> -> <e>*6 -> <e>*<e>*6 -> <e>*<d>*6 -> <e>*3*6 -> <d>*3*6 -> 2*3*6

1. (c) two different parse trees for “2*3*6”

For (a):

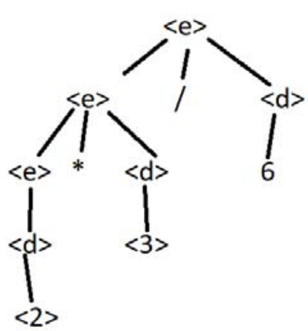


For (b):



1. (d) new grammar – left associative operators

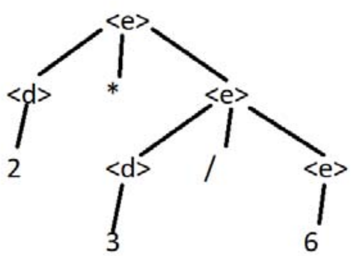
<e> -> <d> | <e>*<d> | <e>/<d>
<d> -> 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9



There are two choices to begin parsing 2*3*6:
(1) <e> -> <e>*<d> or (2) <e> -> <e>/<d>
For option (1), it is impossible to get the expression 2*3/6 from <e> -> <e>*<d> -> <e>*6.
For option (2), it is shown above. Hence, (2) is the only choice.

1. (e) new grammar – right associative operators

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



2. An unambiguous BNF grammar

<e> -> <e>,<n> | <n>

<n> -> <n><d> | <d>

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

Left associativity is enforced on “,” through left recursion; rules of <n> also use left recursion. These left-recursive rules remove ambiguity.

3. New grammar for “+” and “-” left-associativity. Precedence of “~” is higher than “+” and “-”

<e> -> <t> | <e> + <t> | <e> - <t>

<t> -> <d> | ~ <t>

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

4. One BNF grammar is as follows:

<email> := <account> @ <subdomains> . <topdomain>

<account> := <letter> | <account><letter> | <account><digit>

<subdomains> := <letOrDigSeq> | <letOrDigSeq> . <subdomains>

<letOrDigSeq> := <letter><letOrDigSeq> | <digit><letOrDigSeq>
| <letter> | <digit>

<topdomain> := edu | org | com

<letter> -> a | b | c | ... | z | A | B | C | ... | Z

<digit> -> 0 | 1 | 2 | ... | 9

5. New grammar

<canonical-num> -> <digit> | <non-zero-digit> <num>

<num> -> <digit> | <digit> <num>

<non-zero-digit> -> 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

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

6. equivalent BNF grammar

<expr> -> -<int> | <int>| -<int>.<int> | <int>.<int>

<int> -> <digit> | <digit><int>

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