Saulo Almeda Nieves 841-13-0170

UNIVERSITY OF PUERTO RICO AT BAYAMON

DEPARTMENT OF COMPUTER SCIENCE

**COTI 4039 – COMPARATIVE PROGRAMMING LANGUAGES**

**ASSIGNMENT #1 – 47 points**

1. (8 points) Consider the following grammar:

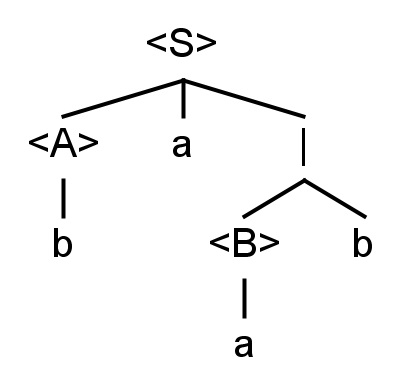
<S> ::= <A>**a**<B>**b**

<A> ::= <A>**b** | **b**

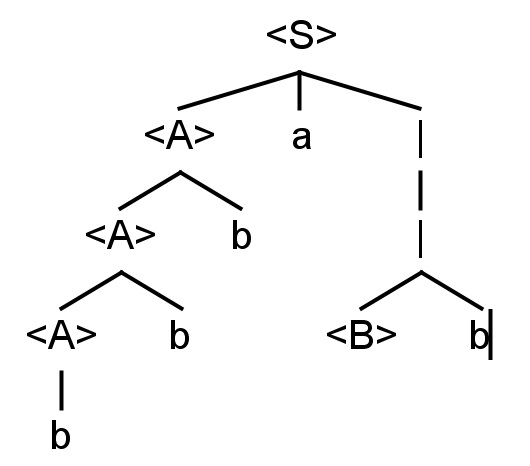
<B> ::= **a**<B> | **a**

Which of the following strings are in the language generated by this grammar? Prove your answer by drawing the parse tree for each string

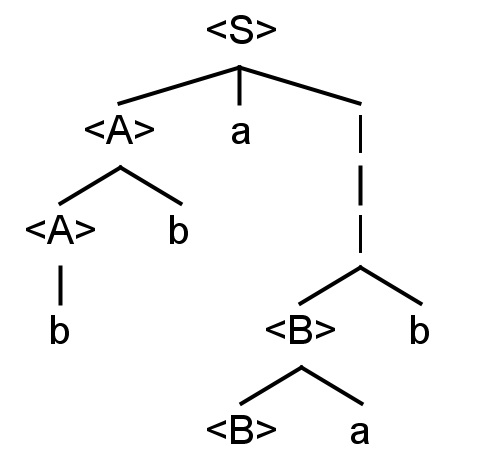
1. baab



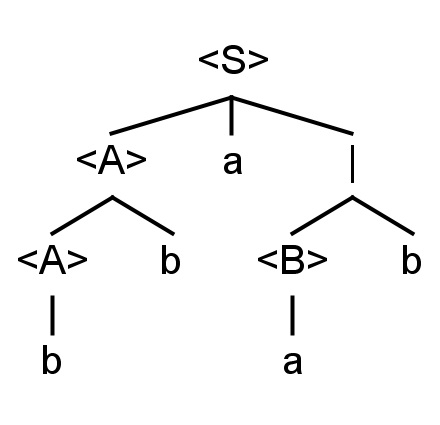
1. bbbab



1. bbaaaaa



1. bbaab



1. (5 points) Give context-free grammars using Backus-Naur Form (BNF) for real numbers in which either the integer part or the fractional part can be empty, but not both. Thus, the grammar must allow **31.**, **3.1**, and **.13**, but not a decimal point by itself (in other words, **.** is **not** a real number).

<real-numbers> ::= <integer-part> ‘.’ <fractional-part> | ‘.’ <fractional-part> | <integer-part> ‘.’

<integer-part> ::= <integer-part> <digit> | <digit>

<fractional-part> ::= <digit> <fractional-part> | <digit>

<digit> ::= ‘0’ | ‘1’ | ‘2’ | ‘3’ | ‘4’ | ‘5’ | ‘6’ | ‘7’ | ‘8’ | ‘9’

1. (5 points) Give context-free grammars using Extended Backus-Naur Form (EBNF) for the real numbers defined in exercise #2.

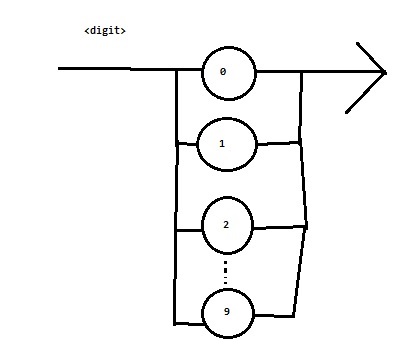
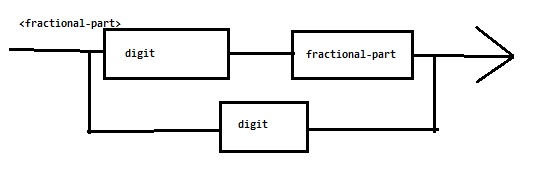
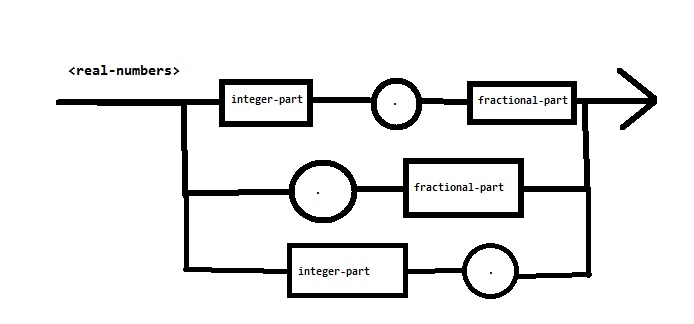
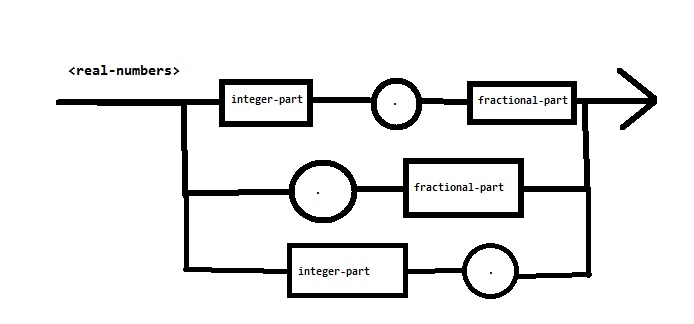
<real-numbers> ::= <integer-part> ‘.’ <fractional-part> | ‘.’ <fractional-part> | <integer-part> ‘.’

<integer-part> ::= { <digit> }

<fractional-part> ::= { <digit> }

<digit> ::= ‘0’ | ‘1’ | ‘2’ | … | ‘9’

1. (5 points) Draw the syntax diagrams for the grammar defined in exercise #2.



1. (8 points) The following grammar describes expressions in a language:

<*exp*> ::= <*exp*> **+** <*mulexp*> | <*mulexp*>

<*mulexp*> ::= <*mulexp*> **\*** <*rootexp*> | <*rootexp*>

<*rootexp*> ::= **(** <*exp*> **)** | **a** | **b** | **c**

Modify this grammar to:

* 1. Add a left-associative / operator, at the same precedence than the \* operator.
  2. Add a right-associative \*\* operator, at higher precedence than the \* and / operators.
  3. Add a left-associative – operator, at the same precedence than the + operator.
  4. Add a left-associative % operator, at higher precedence than the + and – operators but lower precedence than the \* and / operators.

1. (8 points) Rewrite the following expressions in prefix and postfix notation.
   1. a\* b + c / d

Prefix: +\*a b/c d

Postfix: a b\*c d/+

* 1. a \* (b + c) / d

Prefix: /\*a+b c d

Postfix: a b c+\*d/

* 1. (a \* b + c) / d

Prefix: /+\*a b c d

Postfix: a b\*c+d/

* 1. a \* ((b + c) / d)

Prefix: \*a/+b c d

Postfix: a b c+d/\*

1. (8 points) Draw the abstract syntax trees for the expressions in exercise #6.

