CS 3520 - Programming Language Structures

Program: 2
Points: 40
Due Date: October 18 11:59 pm

Write a Ruby program to create a generator for the language defined in the given Context Free Grammar (CFG) and generate all sentences that are less than or equal to a given length. The grammar and the length will be input. There will be a blank line between the rules of the CFG and the length. Each rule is on one line. There will be exactly one space separating all terminals, nonterminals and the arrow in a rule.

You must run your program on the sample input given as well as on several sets of your own input data. Your program should print out the number of nonterminals, the number of rules, echo the inputted rules, and follow this with a list of legal sentences that are less than or equal to the inputted length. See sample input and output below. You will have a main driver in prog2.rb. You will implement the generator in a class named CFGrammar. You must submit your two files, context_free_grammar.rb and prog2.rb to the Grader. When you submit to the Grader, your program must read from the standard input.

The CFGrammar class

```
#Document this file as described here and in the syllabus.

class CFGrammar

   def initialize
     @cfg_rules = Array.new
     @count_rules = 0
     @max_len = 0
     @hash_words = Hash.new
     @nonterminals = Array.new
   end

   # read in rules and the last 2 lines
   # and chomp off the newline character. Check out method chomp
   def read_rules
     while line = gets do
       # store chomped line into @cfg_rules
     end

   # put last value of @cfg_rules into @max_len and
   # remove last two elements of @cfg_rules

   end # read_rules
```

Program Requirements:

You must adhere to the following (loss of points for non-compliance):

1. You may assume that all nonterminals are not substrings of other nonterminals.
   You may assume that all terminals are not substrings of nonterminals.
2. You may want to follow the algorithm hinted here:
   Create an array of rules
   Create a hash-table for each nonterminal (as key) and their right-hand sides (as value) (if more than one right-hand side, group them into an array)
   Create an array of nonterminals
   Get number of rules and print rules
   Generate sentences
      Push start symbol on stack
      While stack is NOT empty
         pop an element off stack
         get length of words (number of words in the element just popped)
         if length <= maxlen, handle/generate via derivation
            loop to find the first (leftmost) nonterminal in what was popped off stack
            if found one nonterminal
               loop to **replace** the found nonterminal with its right-hand sides on stack
            else
               puts the sentence
3. You may assume that the first nonterminal encountered is the start symbol.
4. You must follow the format of the Sample Input and Sample Output given below.

Additional Notes:

- This is an individual assignment, and you must complete it by yourself. Moderate help from others is accepted, but you must make sure you understand the concepts.
- You must comment your code appropriately.
- You will lose points if your final submission is after the due time:
  - -5 if your final submission is by 11:59 PM, October 18
  - -10 if your final submission is by 11:59 PM, October 19
  - -15 if your final submission is by 11:59 PM, October 20
  - No credit after 11:59 PM, October 20
- For all programs in this class, you must turn in a running solution in order to pass the course. The solutions do not have to work correctly for all test cases but must solve a significant portion of the problem.
- Match the output exactly and follow the Ground Rules below.

Ground rules for this program:

1. Variable and method names must be in lowercase with underscores between words in a name
2. Use spaces between operators
3. Comment at the top of each file saying what the class does and your name as the Author
4. One- or two-line comment for each method
5. Methods must be less than 40 lines long
6. Indent **two** spaces in all the normal places (if, while, for, etc.)
7. Line length MUST be 78 characters or less

0.5 points off for each violation. Maximum 4 points off for formatting errors.

Sample Input:

```
program -> statements
statements -> singleStatement ;
statements -> singleStatement ; statements
singleStatement -> variable = expression
expression -> term
expression -> term + expression
expression -> term - expression
term -> factor
term -> factor * term
term -> factor / term
factor -> variable
factor -> const
variable -> a

6
```

Corresponding Sample Output:

```
There are 7 nonterminal symbols.
There are 13 rules in the CFG that follows.

program -> statements
statements -> singleStatement ;
statements -> singleStatement ; statements
singleStatement -> variable = expression
expression -> term
expression -> term + expression
expression -> term - expression
term -> factor
term -> factor * term
term -> factor / term
```

```
factor -> variable
factor -> const
variable -> a

Sentences of length 6 or less are:

a = const - const ;
a = const - a ;
a = a - const ;
a = a - a ;
a = const + const ;
a = const + a ;
a = a + const ;
a = a + a ;
a = const / const ;
a = const / a ;
a = a / const ;
a = a / a ;
a = const * const ;
a = const * a ;
a = a * const ;
a = a * a ;
a = const ;
a = a ;
```