Test 1

Programming Language Concepts

October 2, 2020

1. (10 points) Rewrite the following rule base as a CFG and provide its formal definition (S is the start state)

   $S \rightarrow aSb \mid bAA$

   $A \rightarrow b\,\{aB\} \mid a \mid Bc$

   $B \rightarrow aB \mid c$

2. (10 points) Generate the 10 smallest possible strings using the above rule base. (from problem 1)

3. (10 points) Either show that the following strings are in this lanaguage or state why these strings can not be..

   $S \rightarrow AB \mid BC \mid cB$

   $A \rightarrow Ac \mid aBB \mid b$

   $B \rightarrow Cb \mid cBb \mid a$

   $C \rightarrow Ac \mid bCA \mid c$

   a) aCbcBbcBb b) ccbcbbb b) cabbCA

4. (10 points) Generate the 10 smallest possible strings using the following rule base. If a word doesn't have a described rule treat it as a TERMINAL SYMBOL and count it as ONE CHARACTER.

```
ForStmt = "for" [ Condition | ForClause | RangeClause ] Block
Condition = Expression
RangeClause = [ ExpressionList "=" | IdentifierList ":=" ] "range" Expression
ForClause = [ InitStmt ] ";" [ Condition ] ";" [ PostStmt ]
InitStmt = SimpleStmt
PostStmt = SimpleStmt
SimpleStmt = EmptyStmt | ExpressionStmt | SendStmt | IncDecStmt | Assignment | ShortVarD
Expression = UnaryExpr | Expression binary_op Expression
UnaryExpr  = PrimaryExpr | unary_op UnaryExpr
binary_op  = "||" | "&&" | rel_op | add_op | mul_op
unary_op   = "+" | "-" | "!" | "^" | "*" | "&" | "<-"
Block = "{" StatementList "}"
StatementList = { Statement ";" }
IdentifierList = identifier { "," identifier }
ExpressionList = Expression { "," Expression }
```

5. (10 points) Convert the previous eBNF to a CFG and provide the formal definition of it.

6. (10 points) Find the weakest precondition:

```
if (x > y)
   y = 2 * x + 1
else
   y = 3 * x - 1;
a = x / ( y / 3 );
{a must be a positive integer}
```

7. (10 points) Find the weakest precondition:

```
if (x > y)
   y = 2 * x + 1
else
   y = 3 * x - 1;
a = x / ( y / 3 );
{a must be a positive integer}
```

8. (10 points) Prove total correctness of the following Loop:

```
{some_num > 0}
i = some_num;
apps = 0;
while( i != 0 ){
   apps = apps + i;
   --i;
}
{apps = 1 + 2 + . . . + some_num}
```

9. (10 points) Prove the correctness of the following:

```
{x != 0}
i = x / x;
if ( x < 0 )
    value = ( -x );
else
    value = x;
temp = value;
value = i;
i = temp;
while( i > 0 ){
   value *= i--;
}
{value = x!}
```


_____ 40 points

10. (10 points) Draw and decorate the parse tree for the following Attribute Grammar for the following statement:

    $word = 2.0 * (5 - 10)$

    *** Assign is your STARTING SYMBOL

```
Assign =: identifier = Expr
Expr =: Expr + Term | Expr - Term | Term
Term =: Term * Factor | Term / Factor | Factor
Factor  =:  "(" Expr ")" | integer | float | identifier

Assign =: identifier = Expr [ identifier.value <= Expr.value  ]
Assign =: identifier = Expr [ identifier.actual_type ==> Expr.expected_type ]
Expr1 =: Expr2 + Term [ Expr1.value = Expr2.value * Term.value ]
Expr1 =: Expr2 + Term [ Expr1.type <==
          if (Expr2.type == Term.type == integer)  then  integer  else float  ]
Expr1 =: Expr2 - Term [ Expr1.value = Expr2.value + Term.value ]
Expr1 =: Expr2 - Term [ Expr1.type <==
          if (Expr2.type == Term.type == integer)  then  integer  else float  ]
Expr =: Term [ Expr.value = Term.value ]
Expr =: Term [ Expr.type = Term.type ]
Term1 =: Term2 * Factor [ Term1.value = Term2.value / Factor.value ]
Term1 =: Term2 * Factor [ Term1.type <==
          if (Term2.type == Factor.type == integer) - then  integer  else float  ]
Term1 =: Term2 / Factor [ Term1.value = Term2.value  Factor.value ]
Term1 =: Term2 / Factor [ Term1.type <==
          if (Term2.type == Factor.type == integer)  then  integer  else float  ]
Term =: Factor [ Term.value = Factor.value ]
Term =: Factor [ Term.type = Factor.type ]
Factor =: "(" Expr ")" [ Factor.value =  Expr.value ]
Factor =: integer [ Factor.value = strToInt(integer.str) ]
Factor =: float [ Factor.value = strToFloat(float.str) ]
Factor =: identifier [ Factor.value = VARMAP(identifier.str) ]
```

11. (10 points) Rewrite the following rule in the link provided as an EBNF with the rules provided in chapter 3 and as a Context Free Grammar.

    https://en.cppreference.com/w/cpp/language/floating_literal

12. (10 points) Choose an Esoteric programming language and write a lexical analyzer for it in the language of your choice.