G1.txt

N = S A

E = a b c d

S

S -> aAd

A -> b | bA | Ac

G2.txt

program compound_stmt stmt simple_stmt struct_stmt declaration_stmt type type_ array_declaration assignment_stmt expression term factor io_stmt read_stmt write_stmt ruct_stmt if_stmt condition relation for_stmt

number char string begin end for if else cwrite cread < > <= == >= <> = * + - / [ ] ( ) { } ;

program

program -> "begin" compound_stmt "end"

compound_stmt -> stmt | stmt compound_stmt

stmt -> simple_stmt | struct_stmt

simple_stmt -> declaration_stmt | assignment_stmt | io_stmt

declaration_stmt -> type identifier ";"

type -> type_ | array_declaration

type_ -> "number" | "char" | "string"

array_declaration -> type_ "[" const_int "]"

assignment_stmt -> identifier "=" expression ";"

expression -> expression "+" term | expression "-" term | term

term -> term "*" factor | term "/" factor | factor

factor -> "(" expression ")" | identifier | const_int | const_string

io_stmt -> read_stmt | write_stmt

read_stmt -> "cread(" identifier ");"

write_stmt -> "cwrite" "(" identifier ");" | "cwrite" "(" const_int | const_string ");"

struct_stmt -> compound_stmt | if_stmt | for_stmt

if_stmt -> "if" condition stmt | "if" "(" condition ")" { stmt } "else" { stmt }

condition -> expression relation expression

relation -> "<" | "<=" | "==" | "<>" | ">=" | ">"

for_stmt -> "for" "(" assignment_stmt ";" condition ";" expression ")" "{" stmt "}"

Grammar.py

```
class Grammar:
    def __init__(self, input_filename):
        self.__filename = input_filename
        self.__read_grammar()
        self.__set_of_non_terminals = self.__grammar['non_terminals']
        self.__set_of_terminals = self.__grammar['terminals']
        self.__starting_symbol = self.__grammar['starting_symbol']
        self.__productions = self.__grammar['starting_symbol']

    def __read_grammar(self) -> dict:
        self.__grammar = dict()
        with open(self.__filename) as file:
            # non terminals
            line = file.readline()
            self.__grammar['non_terminals'] = line.strip().split()
            # terminals
            line = file.readline()
            self.__grammar['terminals'] = line.strip().split()
            # starting symbol
            line = file.readline()
            self.__grammar['starting_symbol'] = line.strip()
            # productions
            self.__grammar['productions'] = dict()
            line = file.readline()
```

```
        while line:
            line = line.strip().split('->')
            left_hand_side = line[0].strip()
            right_hand_side = line[1].strip().split('|')
            self.__grammar['productions'][left_hand_side] = list()
            for production in right_hand_side:
                self.__grammar['productions'][left_hand_side].append(production.strip())
            line = file.readline()

def get_grammar(self):
    return self.__grammar
```