UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO

Trabalho Prático - parte A

SSC-0205 Linguagens Formais

Danilo Zecchin Nery	8602430
Frederico de Azevedo Marques	8936926
Roberto Pommella Alegro	8936756

Introdução:

Esse trabalho descreve descreve nas formas BNF (Backus-Naur Form) e EBNF (Extended Backus-Naur Form) a linguagem BC (arbitrary-precision arithmetic language and calculator) no padrão POSIX. A primeira parte consiste a descrição na forma BNF a partir da documentação do FreedBSD. A segunda parte deste trabalho descreve a cadeia de derivação de alguns programas com intuito de demonstrar a aplicabilidade do formalismo descrito na primeira parte. Por fim, a terceira parte contém a descrição da mesma linguagem na forma EBNF.

Forma BNF

```
G = { Vn, Vt, P, <program> };

Vn = {
    <NUMBER>, <INTEGER>, <program>, <input_item>, <break_list>,
    <function_def>, <statement>, <opt_args_list>, <opt_auto_defs_list>,
    <statement_list>, <args_list>, <defs_list>, <expression>, <STRING>,
    <named_var>, <relational>, <opt_exp_list>, <assignment>,
    <plus_minus>, <mult>, <power>, <minus>, <unary_minus>, <inc_dec>
}

Vt = {
    <NL>, <PLUS>, <MINUS>, <POWER>, <INC_DEC>, <MULT_OP>, <RELATION>, <ASSIGNMENT>,
<DIGIT>, <LETTER>, <EOF>
}

P = {

<NL>              ::= ‘\n’
<PLUS>            ::= ‘+’
<MINUS>           ::= ‘-’
<POWER>           ::= ‘^’
<INC_DEC>         ::= ‘++’ | ‘--’
<MULT_OP>         ::= ‘*’  | ‘/’  | ’%’
<RELATION>        ::= ‘==’ | ‘<=’ | ‘>=’ | ’!=’ | ’<’  | ’>’
<ASSIGNMENT>      ::= ‘=’  | ‘+=’ | ’-=’ | ’*=’ | ’/=’ | ’%=’ | ’^=’

<NUMBER> ::= <INTEGER>
      | <INTEGER> ‘.’
      | ‘.’ <INTEGER>
      | <INTEGER> ‘.’ <INTEGER>

<INTEGER> ::= <DIGIT>
      | <INTEGER> <DIGIT>


<DIGIT> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F

<LETTER> ::= a | b | c | d | e | f | g | h | i | j | k | l | m
           | n | o | p | q | r | s | t | u | v | w | x | y | z
```

```
<program> ::= <EOF>
      | <input_item> <program>

<input_item> ::= <break_list> <NL>
      | <function_def>

<break_list> ::= // empty
      | <statement>
      | <break_list> ‘;’
      | <break_list> ‘;’ <statement>

<function_def> ::= ‘define’ <LETTER> ‘(‘ <opt_args_list> ‘)’ ‘{‘ <NL>
               <opt_auto_defs_list>
               <statement_list>
               ‘}’

<opt_args_list> ::= // empty
      | <args_list>

<args_list> ::= <LETTER>
      | <args_list> ‘,’ <LETTER>


<opt_auto_defs_list> ::= // empty
      | ‘auto’ <defs_list> <NL>
      | ‘auto’ <defs_list> ‘;’


<defs_list> ::= <LETTER>
      | <LETTER> ‘[‘ ‘]’
      | ‘defs_list’, <LETTER>
      | ‘defs_list’, <LETTER> ‘[‘ ‘]’

<statement_list> ::= // empty
      | <statement>
      | <statement_list> <NL>
      | <statement_list> ‘;’
      | <statement_list> <NL> <statement>
      | <statement_list> ‘;’ <statement>

<statement> ::= <expression>
      | ‘{‘ <statement_list> ‘}’
      | ‘if’ ‘(‘<expression>‘)’<statement>
      | ‘while’ ‘(‘<expression>‘)’<statement>
      | ‘for’ ‘(‘<expression>’;’<expression>’;’<expression>’)’<statement>
      | <STRING>
```

```
      | ‘quit’
      | ‘break’
      | ‘return’
      | ‘return’ ‘(‘ <expression> ‘)’

<expression> ::= <NUMBER>
      | <named_var>
      | <relational>
      | ‘(‘ <expression> ‘)’
      | LETTER ‘(‘ <opt_exp_list> ‘)’
      | ‘length’ ‘(‘ <expression> ‘)’
      | ‘scale’ ‘(‘ <expression> ‘)’
      | ‘sqrt’ ‘(‘ <expression> ‘)’

<opt_exp_list> ::= // empty
      | <expression>
      | <opt_exp_list> ‘,’ <expression>

<relational> ::= <assignment> <RELATION> <relational>
      | <assignment>

<assignment> ::= <named_var> <ASSIGNMENT> <assignment>
      | <plus_minus>

<plus_minus> ::= <mult> <PLUS> <plus_minus>
      | <mult> <MINUS> <plus_minus>
      | <mult>

<mult> ::= <power> <MULT_OP> <mult>
      | <power>

<power> ::= <minus> <POWER> <power>
      | <unary_minus>

<unary_minus> ::= <MINUS> <inc_dec>
      | <inc_dec>

<inc_dec> ::= <INC_DEC> <named_var>
      | <named_var> <INC_DEC>
      | <expression>

<named_var> ::= <LETTER>
      | <LETTER> ‘[‘ <expression> ‘]’
      | ‘scale’
      | ‘ibase’
      | ‘obase’
```

Árvore de derivação

Derivação de programa: Função fatorial

```
BEGIN SNIPPET

define f(n) {
  if (n == 0) return (1)
  return (n * f(n-1))
}

END SNIPPET
```

1. <program>
2. <input_item> <program>
3. <function_def> <program>
4. 'define'<LETTER>'(' <opt_args_list> ')' '{'<NL><opt_auto_defs_list> <statement_list> '}' <program>
5. 'define' 'f' '(' <opt_args_list> ')' '{' <NL> <opt_auto_defs_list> <statement_list> '}' <program>
6. 'define' 'f' '(' <args_list> ')' '{' <NL> <opt_auto_defs_list> <statement_list> '}' <program>
7. 'define' 'f' '(' <LETTER> ')' '{' <NL> <opt_auto_defs_list> <statement_list> '}' <program>
8. 'define' 'f' '(' 'n' ')' '{' <NL> <opt_auto_defs_list> <statement_list> '}' <program>
9. 'define' 'f' '(' 'n' ')' '{' '\n'
   <opt_auto_defs_list> <statement_list> '}' <program>
10. 'define' 'f' '(' 'n' ')' '{' '\n'
   '' <statement_list> '}' <program>
11. 'define' 'f' '(' 'n' ')' '{' '\n'
   <statement_list> <NL> <statement> '}' <program>
12. 'define' 'f' '(' 'n' ')' '{' '\n'
   <statement> <NL> <statement> '}' <program>
13. 'define' 'f' '(' 'n' ')' '{' '\n'
   'if' '(' <expression> ')' <statement> <NL> <statement> '}' <program>
14. 'define' 'f' '(' 'n' ')' '{' '\n'
   'if' '(' <relational> ')' <statement> <NL> <statement> '}' <program>
15. 'define' 'f' '(' 'n' ')' '{' '\n'
   'if' '(' <assignment> <RELATION> <relational> ')' <statement> <NL> <statement> '}' <program>
16. 'define' 'f' '(' 'n' ')' '{' '\n'
   'if' '(' <plus_minus> <RELATION> <relational> ')' <statement> <NL> <statement> '}' <program>
17. 'define' 'f' '(' 'n' ')' '{' '\n'
   'if' '(' <mult> <RELATION> <relational> ')' <statement> <NL> <statement> '}' <program>
18. 'define' 'f' '(' 'n' ')' '{' '\n'

'if' '(' <power> <RELATION> <relational> ')' <statement> <NL> <statement> '}' <program>

19. 'define' 'f' '(' 'n' ')' '{' '\n'
'if' '(' <unary_minus> <RELATION> <relational> ')' <statement> <NL> <statement> '}' <program>

20. 'define' 'f' '(' 'n' ')' '{' '\n'
'if' '(' <inc_dec> <RELATION> <relational> ')' <statement> <NL> <statement> '}' <program>

21. 'define' 'f' '(' 'n' ')' '{' '\n'
'if' '(' <expression> <RELATION> <relational> ')' <statement> <NL> <statement> '}' <program>

22. 'define' 'f' '(' 'n' ')' '{' '\n'
'if' '(' <named_var> <RELATION> <relational> ')' <statement> <NL> <statement> '}' <program>

23. 'define' 'f' '(' 'n' ')' '{' '\n'
'if' '(' 'n' <RELATION> <relational> ')' <statement> <NL> <statement> '}' <program>

24. 'define' 'f' '(' 'n' ')' '{' '\n'
'if' '(' 'n' '==' <relational> ')' <statement> <NL> <statement> '}' <program>

25. 'define' 'f' '(' 'n' ')' '{' '\n'
'if' '(' 'n' '==' <assignment> ')' <statement> <NL> <statement> '}' <program>

26. 'define' 'f' '(' 'n' ')' '{' '\n'
'if' '(' 'n' '==' <plus_minus> ')' <statement> <NL> <statement> '}' <program>

27. 'define' 'f' '(' 'n' ')' '{' '\n'
'if' '(' 'n' '==' <mult> ')' <statement> <NL> <statement> '}' <program>

28. 'define' 'f' '(' 'n' ')' '{' '\n'
'if' '(' 'n' '==' <power> ')' <statement> <NL> <statement> '}' <program>

29. 'define' 'f' '(' 'n' ')' '{' '\n'
'if' '(' 'n' '==' <unary_minus> ')' <statement> <NL> <statement> '}' <program>

30. 'define' 'f' '(' 'n' ')' '{' '\n'
'if' '(' 'n' '==' <inc_dec> ')' <statement> <NL> <statement> '}' <program>

31. 'define' 'f' '(' 'n' ')' '{' '\n'
'if' '(' 'n' '==' <expression> ')' <statement> <NL> <statement> '}' <program>

32. 'define' 'f' '(' 'n' ')' '{' '\n'
'if' '(' 'n' '==' <NUMBER> ')' <statement> <NL> <statement> '}' <program>

33. 'define' 'f' '(' 'n' ')' '{' '\n'
'if' '(' 'n' '==' '0' ')' <statement> <NL> <statement> '}' <program>

34. 'define' 'f' '(' 'n' ')' '{' '\n'
'if' '(' 'n' '==' '0' ')' 'return' '(' <expression> ')' <NL> <statement> '}' <program>

35. 'define' 'f' '(' 'n' ')' '{' '\n'
'if' '(' 'n' '==' '0' ')' 'return' '(' <NUMBER> ')' <NL> <statement> '}' <program>

36. 'define' 'f' '(' 'n' ')' '{' '\n'
'if' '(' 'n' '==' '0' ')' 'return' '(' '1' ')' <NL> <statement> '}' <program>

37. 'define' 'f' '(' 'n' ')' '{' '\n'
'if' '(' 'n' '==' '0' ')' 'return' '(' '1' ')' '\n'
<statement> '}' <program>

38. 'define' 'f' '(' 'n' ')' '{' '\n'

'if' '(' 'n' '==' '0' ')' 'return' '(' '1' ')' '\n'
'return' '(' <expression> ')' '}' <program>

39. 'define' 'f' '(' 'n' ')' '{' '\n'
'if' '(' 'n' '==' '0' ')' 'return' '(' '1' ')' '\n'
'return' '(' <relational> ')' '}' <program>

40. 'define' 'f' '(' 'n' ')' '{' '\n'
'if' '(' 'n' '==' '0' ')' 'return' '(' '1' ')' '\n'
'return' '(' <assignment> ')' '}' <program>

41. 'define' 'f' '(' 'n' ')' '{' '\n'
'if' '(' 'n' '==' '0' ')' 'return' '(' '1' ')' '\n'
'return' '(' <plus_minus> ')' '}' <program>

42. 'define' 'f' '(' 'n' ')' '{' '\n'
'if' '(' 'n' '==' '0' ')' 'return' '(' '1' ')' '\n'
'return' '(' <mult> ')' '}' <program>

43. 'define' 'f' '(' 'n' ')' '{' '\n'
'if' '(' 'n' '==' '0' ')' 'return' '(' '1' ')' '\n'
'return' '(' <power> <MULT_OP> <mult> ')' '}' <program>

44. 'define' 'f' '(' 'n' ')' '{' '\n'
'if' '(' 'n' '==' '0' ')' 'return' '(' '1' ')' '\n'
'return' '(' <unary_minus> <MULT_OP> <mult> ')' '}' <program>

45. 'define' 'f' '(' 'n' ')' '{' '\n'
'if' '(' 'n' '==' '0' ')' 'return' '(' '1' ')' '\n'
'return' '(' <inc_dec> <MULT_OP> <mult> ')' '}' <program>

46. 'define' 'f' '(' 'n' ')' '{' '\n'
'if' '(' 'n' '==' '0' ')' 'return' '(' '1' ')' '\n'
'return' '(' <named_var> <MULT_OP> <mult> ')' '}' <program>

47. 'define' 'f' '(' 'n' ')' '{' '\n'
'if' '(' 'n' '==' '0' ')' 'return' '(' '1' ')' '\n'
'return' '(' <LETTER> <MULT_OP> <mult> ')' '}' <program>

48. 'define' 'f' '(' 'n' ')' '{' '\n'
'if' '(' 'n' '==' '0' ')' 'return' '(' '1' ')' '\n'
'return' '(' 'n' <MULT_OP> <mult> ')' '}' <program>

49. 'define' 'f' '(' 'n' ')' '{' '\n'
'if' '(' 'n' '==' '0' ')' 'return' '(' '1' ')' '\n'
'return' '(' 'n' '*' <mult> ')' '}' <program>

50. 'define' 'f' '(' 'n' ')' '{' '\n'
'if' '(' 'n' '==' '0' ')' 'return' '(' '1' ')' '\n'
'return' '(' 'n' '*' <power> ')' '}' <program>

51. 'define' 'f' '(' 'n' ')' '{' '\n'
'if' '(' 'n' '==' '0' ')' 'return' '(' '1' ')' '\n'
'return' '(' 'n' '*' <unary_minus> ')' '}' <program>

52. 'define' 'f' '(' 'n' ')' '{' '\n'

```
   'if' '(' 'n' '==' '0' ')' 'return' '(' '1' ')' '\n'
   'return' '(' 'n' '*' <inc_dec> ')' '}' <program>
53. 'define' 'f' '(' 'n' ')' '{' '\n'
   'if' '(' 'n' '==' '0' ')' 'return' '(' '1' ')' '\n'
   'return' '(' 'n' '*' <expression> ')' '}' <program>
54. 'define' 'f' '(' 'n' ')' '{' '\n'
   'if' '(' 'n' '==' '0' ')' 'return' '(' '1' ')' '\n'
   'return' '(' 'n' '*' <LETTER> '(' <opt_exp_list> ')' ')' '}' <program>
55. 'define' 'f' '(' 'n' ')' '{' '\n'
   'if' '(' 'n' '==' '0' ')' 'return' '(' '1' ')' '\n'
   'return' '(' 'n' '*' 'f' '(' <opt_exp_list> ')' ')' '}' <program>
56. 'define' 'f' '(' 'n' ')' '{' '\n'
   'if' '(' 'n' '==' '0' ')' 'return' '(' '1' ')' '\n'
   'return' '(' 'n' '*' 'f' '(' <expression> ')' ')' '}' <program>
57. 'define' 'f' '(' 'n' ')' '{' '\n'
   'if' '(' 'n' '==' '0' ')' 'return' '(' '1' ')' '\n'
   'return' '(' 'n' '*' 'f' '(' <relational> ')' ')' '}' <program>
58. 'define' 'f' '(' 'n' ')' '{' '\n'
   'if' '(' 'n' '==' '0' ')' 'return' '(' '1' ')' '\n'
   'return' '(' 'n' '*' 'f' '(' <plus_minus> ')' ')' '}' <program>
59. 'define' 'f' '(' 'n' ')' '{' '\n'
   'if' '(' 'n' '==' '0' ')' 'return' '(' '1' ')' '\n'
   'return' '(' 'n' '*' 'f' '(' <mult> <MINUS> <plus_minus> ')' ')' '}' <program>
60. 'define' 'f' '(' 'n' ')' '{' '\n'
   'if' '(' 'n' '==' '0' ')' 'return' '(' '1' ')' '\n'
   'return' '(' 'n' '*' 'f' '(' <power> <MINUS> <plus_minus> ')' ')' '}' <program>
61. 'define' 'f' '(' 'n' ')' '{' '\n'
   'if' '(' 'n' '==' '0' ')' 'return' '(' '1' ')' '\n'
   'return' '(' 'n' '*' 'f' '(' <unary_minus> <MINUS> <plus_minus> ')' ')' '}' <program>
62. 'define' 'f' '(' 'n' ')' '{' '\n'
   'if' '(' 'n' '==' '0' ')' 'return' '(' '1' ')' '\n'
   'return' '(' 'n' '*' 'f' '(' <inc_dec> <MINUS> <plus_minus> ')' ')' '}' <program>
63. 'define' 'f' '(' 'n' ')' '{' '\n'
   'if' '(' 'n' '==' '0' ')' 'return' '(' '1' ')' '\n'
   'return' '(' 'n' '*' 'f' '(' <expression> <MINUS> <plus_minus> ')' ')' '}' <program>
64. 'define' 'f' '(' 'n' ')' '{' '\n'
   'if' '(' 'n' '==' '0' ')' 'return' '(' '1' ')' '\n'
   'return' '(' 'n' '*' 'f' '(' <named_var> <MINUS> <plus_minus> ')' ')' '}' <program>
65. 'define' 'f' '(' 'n' ')' '{' '\n'
   'if' '(' 'n' '==' '0' ')' 'return' '(' '1' ')' '\n'
   'return' '(' 'n' '*' 'f' '(' 'n' <MINUS> <plus_minus> ')' ')' '}' <program>
66. 'define' 'f' '(' 'n' ')' '{' '\n'
```

```
   'if' '(' 'n' '==' '0' ')' 'return' '(' '1' ')' '\n'
   'return' '(' 'n' '*' 'f' '(' 'n' '-' <plus_minus> ')' ')' '}' <program>
67. 'define' 'f' '(' 'n' ')' '{' '\n'
   'if' '(' 'n' '==' '0' ')' 'return' '(' '1' ')' '\n'
   'return' '(' 'n' '*' 'f' '(' 'n' '-' <mult> ')' ')' '}' <program>
68. 'define' 'f' '(' 'n' ')' '{' '\n'
   'if' '(' 'n' '==' '0' ')' 'return' '(' '1' ')' '\n'
   'return' '(' 'n' '*' 'f' '(' 'n' '-' <power> ')' ')' '}' <program>
69. 'define' 'f' '(' 'n' ')' '{' '\n'
   'if' '(' 'n' '==' '0' ')' 'return' '(' '1' ')' '\n'
   'return' '(' 'n' '*' 'f' '(' 'n' '-' <unary_minus> ')' ')' '}' <program>
70. 'define' 'f' '(' 'n' ')' '{' '\n'
   'if' '(' 'n' '==' '0' ')' 'return' '(' '1' ')' '\n'
   'return' '(' 'n' '*' 'f' '(' 'n' '-' <inc_dec> ')' ')' '}' <program>
71. 'define' 'f' '(' 'n' ')' '{' '\n'
   'if' '(' 'n' '==' '0' ')' 'return' '(' '1' ')' '\n'
   'return' '(' 'n' '*' 'f' '(' 'n' '-' <expression> ')' ')' '}' <program>
72. 'define' 'f' '(' 'n' ')' '{' '\n'
   'if' '(' 'n' '==' '0' ')' 'return' '(' '1' ')' '\n'
   'return' '(' 'n' '*' 'f' '(' 'n' '-' <NUMBER> ')' ')' '}' <program>
73. 'define' 'f' '(' 'n' ')' '{' '\n'
   'if' '(' 'n' '==' '0' ')' 'return' '(' '1' ')' '\n'
   'return' '(' 'n' '*' 'f' '(' 'n' '-' '1' ')' ')' '}' <program>
74. 'define' 'f' '(' 'n' ')' '{' '\n'
   'if' '(' 'n' '==' '0' ')' 'return' '(' '1' ')' '\n'
   'return' '(' 'n' '*' 'f' '(' 'n' '-' '1' ')' ')' '}' <EOF>
75. 'define' 'f' '(' 'n' ')' '{' '\n'
   'if' '(' 'n' '==' '0' ')' 'return' '(' '1' ')' '\n'
   'return' '(' 'n' '*' 'f' '(' 'n' '-' '1' ')' ')' '}' ''
```

Derivação de programa: Declaração de função

```
BEGIN SNIPPET
define f() {
    auto a
    a = 2
    return (a)
}
END SNIPPET
1. <program>
2. <input_item><program>
3. <function_def><program>
4. 'define' <LETTER> '(' <opt_args_list> ')' '{' <NL>
    <opt_auto_defs_list>
    <statement_list>
    '}' <program>
5. 'define' 'f' '(' <opt_args_list> ')' '{' <NL>
    <opt_auto_defs_list>
    <statement_list>
    '}' <program>
6. 'define' 'f' '(' '' ')' '{' <NL>
    <opt_auto_defs_list>
    <statement_list>
    '}' <program>
7. 'define' 'f' '(' '' ')' '{' '\n'
    <opt_auto_defs_list>
    <statement_list>
    '}' <program>
8. 'define' 'f' '(' '' ')' '{' '\n'
    'auto' <defs_list> <NL>
    <statement_list>
    '}' <program>
9. 'define' 'f' '(' '' ')' '{' '\n'
    'auto' <LETTER> <NL>
    <statement_list>
```

'}' <program>

10. 'define' 'f' '(' '' ')' '{' '\n'
    'auto' 'a' <NL>
    <statement_list>
    '}' <program>
11. 'define' 'f' '(' '' ')' '{' '\n'
    'auto' 'a' '\n'
    <statement_list>
    '}' <program>
12. 'define' 'f' '(' '' ')' '{' '\n'
    'auto' 'a' '\n'
    <statement_list><NL><statement>
    '}' <program>
13. 'define' 'f' '(' '' ')' '{' '\n'
    'auto' 'a' '\n'
    <statement><NL><statement>
    '}' <program>
14. 'define' 'f' '(' '' ')' '{' '\n'
    'auto' 'a' '\n'
    <expression><NL><statement>
    '}' <program>
15. 'define' 'f' '(' '' ')' '{' '\n'
    'auto' 'a' '\n'
    <relational><NL><statement>
    '}' <program>
16. 'define' 'f' '(' '' ')' '{' '\n'
    'auto' 'a' '\n'
    <assignment><NL><statement>
    '}' <program>
17. 'define' 'f' '(' '' ')' '{' '\n'
    'auto' 'a' '\n'
    <plus_minus><ASSIGNMENT><assignment><NL><statement>
    '}' <program>
18. 'define' 'f' '(' '' ')' '{' '\n'
    'auto' 'a' '\n'
    <mult><ASSIGNMENT><assignment><NL><statement>
    '}' <program>

19. 'define' 'f' '(' '' ')' '{' '\n'
    'auto' 'a' '\n'
    <power><ASSIGNMENT><assignment><NL><statement>
    '}' <program>
20. 'define' 'f' '(' '' ')' '{' '\n'
    'auto' 'a' '\n'
    <unary_minus><ASSIGNMENT><assignment><NL><statement>
    '}' <program>
21. 'define' 'f' '(' '' ')' '{' '\n'
    'auto' 'a' '\n'
    <inc_dec><ASSIGNMENT><assignment><NL><statement>
    '}' <program>
22. 'define' 'f' '(' '' ')' '{' '\n'
    'auto' 'a' '\n'
    <expression><ASSIGNMENT><assignment><NL><statement>
    '}' <program>
23. 'define' 'f' '(' '' ')' '{' '\n'
    'auto' 'a' '\n'
    <named_var><ASSIGNMENT><assignment><NL><statement>
    '}' <program>
24. 'define' 'f' '(' '' ')' '{' '\n'
    'auto' 'a' '\n'
    <LETTER><ASSIGNMENT><assignment><NL><statement>
    '}' <program>
25. 'define' 'f' '(' '' ')' '{' '\n'
    'auto' 'a' '\n'
    'a' <ASSIGNMENT><assignment><NL><statement>
    '}' <program>
26. 'define' 'f' '(' '' ')' '{' '\n'
    'auto' 'a' '\n'
    'a' '=' <assignment><NL><statement>
    '}' <program>
27. 'define' 'f' '(' '' ')' '{' '\n'
    'auto' 'a' '\n'
    'a' '=' <plus_minus><NL><statement>
    '}' <program>
28. 'define' 'f' '(' '' ')' '{' '\n'
    'auto' 'a' '\n'
    'a' '=' <mult><NL><statement>
    '}' <program>
29. 'define' 'f' '(' '' ')' '{' '\n'
    'auto' 'a' '\n'
    'a' '=' <power><NL><statement>
    '}' <program>
30. 'define' 'f' '(' '' ')' '{' '\n'
    'auto' 'a' '\n'

```
    'a' '=' <unary_minus><NL><statement>
    '}' <program>
31. 'define' 'f' '(' '' ')' '{' '\n'
    'auto' 'a' '\n'
    'a' '=' <inc_dec><NL><statement>
    '}' <program>
32. 'define' 'f' '(' '' ')' '{' '\n'
    'auto' 'a' '\n'
    'a' '=' <expression><NL><statement>
    '}' <program>
33. 'define' 'f' '(' '' ')' '{' '\n'
    'auto' 'a' '\n'
    'a' '=' '2'<NL><statement>
    '}' <program>
34. 'define' 'f' '(' '' ')' '{' '\n'
    'auto' 'a' '\n'
    'a' '=' '2' '\n'
    <statement>
    '}' <program>
35. 'define' 'f' '(' '' ')' '{' '\n'
    'auto' 'a' '\n'
    'a' '=' '2' '\n'
    'return' '(' <expression> ')'
    '}' <program>
36. 'define' 'f' '(' '' ')' '{' '\n'
    'auto' 'a' '\n'
    'a' '=' '2' '\n'
    'return' '(' <named_var> ')'
    '}' <program>
37. 'define' 'f' '(' '' ')' '{' '\n'
    'auto' 'a' '\n'
    'a' '=' '2' '\n'
    'return' '(' 'a' ')'
    '}' <program>
38. 'define' 'f' '(' '' ')' '{' '\n'
    'auto' 'a' '\n'
    'a' '=' '2' '\n'
    'return' '(' 'a' ')'
    '}' <EOF>
39. 'define' 'f' '(' '' ')' '{' '\n'
    'auto' 'a' '\n'
    'a' '=' '2' '\n'
    'return' '(' 'a' ')'
    '}' ''
```

Derivação de programa: Declaração de função

BEGIN SNIPPET

```
a = 2
-2 * ++a
```

END SNIPPET

1. <program>
2. <input_item><program>
3. <break_list><NL><program>
4. <statement><NL><program>
5. <expression><NL><program>
6. <relational><NL><program>
7. <assignment><NL><program>
8. <named_var><ASSIGNMENT><assignment><NL><program>
9. 'a'<ASSIGNMENT><assignment><NL><program>
10. 'a' '=' <assignment><NL><program>
11. 'a' '=' <plus_minus><NL><program>
12. 'a' '=' <mult><NL><program>
13. 'a' '=' <power><NL><program>
14. 'a' '=' <unary_minus><NL><program>
15. 'a' '=' <inc_dec><NL><program>
16. 'a' '=' <expression><NL><program>
17. 'a' '=' <NUMBER><NL><program>
18. 'a' '=' '2'<NL><program>
19. 'a' '=' '2' '\n'
      <program>
20. 'a' '=' '2' '\n'
      <program>
21. 'a' '=' '2' '\n'
      <program>

22. 'a' '=' '2' '\n'

```
      <input_item>
      <program>
23. 'a' '=' '2' '\n'
      <break_list><NL>
      <program>
24. 'a' '=' '2' '\n'
      <statement><NL>
      <program>
25. 'a' '=' '2' '\n'
      <expression><NL>
      <program>
26. 'a' '=' '2' '\n'
      <relational><NL>
      <program>
27. 'a' '=' '2' '\n'
      <assignment><NL>
      <program>
28. 'a' '=' '2' '\n'
      <plus_minus><NL>
      <program>
29. 'a' '=' '2' '\n'
      <mult><NL>
      <program>
30. 'a' '=' '2' '\n'
      <power><NL>
      <program>
31. 'a' '=' '2' '\n'
      <unary_minus><NL>
      <program>
32. 'a' '=' '2' '\n'
      <MINUS><inc_dec><NL>
      <program>
33. 'a' '=' '2' '\n'
      '-'<inc_dec><NL>
      <program>
```

```
34. 'a' '=' '2' '\n'
      '-'<expression><NL>
      <program>
35. 'a' '=' '2' '\n'
      '-'<relational><NL>
      <program>
36. 'a' '=' '2' '\n'
      '-'<assignment><NL>
      <program>
37. 'a' '=' '2' '\n'
      '-'<plus_minus><NL>
      <program>
38. 'a' '=' '2' '\n'
      '-'<mult><NL>
      <program>
39. 'a' '=' '2' '\n'
      '-'<power><MULT_OP><mult><NL>
      <program>
40. 'a' '=' '2' '\n'
      '-'<unary_minus><MULT_OP><mult><NL>
      <program>
41. 'a' '=' '2' '\n'
      '-'<inc_dec><MULT_OP><mult><NL>
      <program>
42. 'a' '=' '2' '\n'
      '-'<expression><MULT_OP><mult><NL>
      <program>
43. 'a' '=' '2' '\n'
      '-'<NUMBER><MULT_OP><mult><NL>
      <program>
44. 'a' '=' '2' '\n'
      '-' '2'<MULT_OP><mult><NL>
      <program>
45. 'a' '=' '2' '\n'
      '-' '2' '*' <mult><NL>
```

```
      <program>
46. 'a' '=' '2' '\n'
      '-' '2' '*' <power><NL>
      <program>
47. 'a' '=' '2' '\n'
      '-' '2' '*' <unary_minus><NL>
      <program>
48. 'a' '=' '2' '\n'
      '-' '2' '*' <inc_dec><NL>
      <program>
49. 'a' '=' '2' '\n'
      '-' '2' '*' <INC_DEC><named_var><NL>
      <program>
50. 'a' '=' '2' '\n'
      '-' '2' '*' '++'<named_var><NL>
      <program>
51. 'a' '=' '2' '\n'
      '-' '2' '*' '++'<LETTER><NL>
      <program>
52. 'a' '=' '2' '\n'
      '-' '2' '*' '++' 'a'<NL>
      <program>
53. 'a' '=' '2' '\n'
      '-' '2' '*' '++' 'a' '\n'
      <program>
54. 'a' '=' '2' '\n'
      '-' '2' '*' '++' 'a' '\n'
      <EOF>
55. 'a' '=' '2' '\n'
      '-' '2' '*' '++' 'a' '\n'
      ''
```

Forma EBNF

```
<NL>              = ‘\n’;
<PLUS>            = ‘+’;
<MINUS>           = ‘-’;
<POWER>           = ‘^’;
<INC_DEC>         = ‘++’ | ‘--’;
<MULT_OP>         = ‘*’  | ‘/’  | ’%’;
<RELATION>        = ‘==’ | ‘<=’ | ‘>=’ | ’!=’ | ’<’  | ’>’;
<ASSIGNMENT>            = ‘=’  | ‘+=’ | ’-=’ | ’*=’ | ’/=’ | ’%=’ | ’^=’;

<DIGIT> = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7
        | 8 | 9 | A | B | C | D | E | F;

<LETTER> = a | b | c | d | e | f | g | h | i | j | k | l | m
         | n | o | p | q | r | s | t | u | v | w | x | y | z;

<NUMBER> = ( {<DIGIT>}+[.] |  {<DIGIT>}*,’.’,{<DIGIT>}+);


<function_def> = `define`,<LETTER>,`(`,[<LETTER>,{`,`,<LETTER>}*],`)`,`{`,<NL>
           [{`auto`,<var_def>,{`, `<var_def>}*,<NL>}+]
           (<statement> | {<statement>,(<NL> | `;`)}+)
           `}`;

<var_def> = (<LETTER>,`[`,`]` | <LETTER>);

<statement> = <expression>
      | ‘{‘, (<statement> | {<statement> (<NL> | `;`)}+), ‘}’
      | (if | while), ‘(‘, <expression>, ‘)’, <statement>
      | ‘for’, ‘(‘<expression>, ’;’, <expression>, ’;’, <expression>, ’)’,
<statement>
      | <STRING> | ‘quit’ | ‘break’ | `return` [‘(‘ ,<expression>, ‘)’]
      ;

<named_var> = <LETTER> [‘[‘, <expression>, ‘]’]
      | ‘scale’
      | ‘ibase’
      | ‘obase’
      ;
```

```
<expression> = <NUMBER>
      | <named_var>
      | <relational>
      | ‘(‘, <expression>, ‘)’
      | LETTER ‘(‘, [<expression>, {`,`, <expression>]}*], ‘)’
      | (‘length’ | ‘scale’ | ‘sqrt’), ‘(‘, <expression>, ‘)’
      ;

<relational> = <assignment> [<RELATION> <relational>];

<assignment> = <named_var> <ASSIGNMENT> <assignment>
      | <plus_minus>
      ;

<plus_minus> = <mult> [(<PLUS> | <MINUS>) <plus_minus>];

<mult> = <power> [<MULT_OP> <mult>];

<power> = <minus> <POWER> <power>
      | <unary_minus>
      ;

<unary_minus> = <MINUS> <inc_dec>
      | <inc_dec>
      ;

<inc_dec> = <INC_DEC>, <named_var>
      | <named_var>, <INC_DEC>
      | <expression>
      ;

<named_var> = <LETTER> [‘[‘, <expression>, ‘]’]
      | ‘scale’
      | ‘ibase’
      | ‘obase’
      ;
```