If You Know, You Know

Programming Language Documentation

Authors:

Alamag, Jose Luis
Calendario, Mark Kenneth
Caspe, Mark Vincent
Favorito, Vince Lennard
Lalis, Reygine
Villegas, Daniel

Web App Preview

https://iykyk-31n.vercel.app/

Repository

https://github.com/markcalendario/IYKYK-programming-language

About this Document

This paper is created by the students of BSCS 3-1N Group 2 as a final requirement for the course Principles of Programming Languages. With the guidance of the authors' professor, Mr. Montaigne Molejon, IYKYK programming language, achieved the goal of creating a trendy and GenZ style of writing codes. For more information on how to run the actual programming language and its different parts or to check the official source code, the documentation and source code is published on GitHub (https://github.com/markcalendario/IYKYK-programming-language)

Table of Content

Introduction

Language Background

IYKYK stands for “If You Know, You Know,” and is based on the GenZ term of the same acronym commonly used to express an inside joke in a group of people. Being named after a term used by a generation known for their laid back but witty character, IYKYK uses GenZ terms by relating its meanings to the properties of data types, functions, and keywords in the programming language. The developers used the concept of GenZ terms to make it beginner friendly, specially to the younger learners of programming who may find the terms in other languages intimidating.

The paradigm used by this language is influenced by C and C++, both are commonly used languages. It also pulls out concepts from C# and Javascript. Despite its friendly nature that makes it look simple, it was created to improve and solve some principles in programming related to callback functions, function contractors, safety for undefined values, incrementation, and looping. It also has features that support HTML and CSS coding for web development. These features make IYKYK a powerful and useful language.

Software Features

IYKYK code can be run on its own portal, https://iykyk-31n.vercel.app/. This website allows users to create their own IYKYK programming session or get a session code for collaborative programming. This means there can be more than one programmer in one session. During a session, changes are updated and can be seen live, similar to online document editing softwares. This makes collaboration easier and faster. The website also features a lexical analyzer (lexer), syntax analyzer (parser), save file (export), import, share session code (link), and delete session. A programming language inspired by a generation who grew up with fast communication will surely thrive in an environment where collaborative programming can be done easily.

SYNTACTICAL ELEMENTS

mainCharacter = {characters, numbers, extra}
characters = {highKey, Lowkey}
numbers = {-figure* | figure*}
highKey = {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}
lowKey = {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}
figure = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
extra = {!, {, }, (,), [,], ., ,, <, >, =, +, *, -, /, _, %, \, ^}

GENERAL PRODUCTION RULE

<**IYKYK_PROGRAMMING**> ::= <ROUTINE_DECLARATION>* <PROG_STMT>+

<**PROG_STMT**> ::= <DEC_STMTS> | <INPUT_ STMTS > | <OUT_ STMTS > | <ASSIGN_ STMTS > | <COND_ STMTS > | <LOOP_STMT> | <WEBDEV_STMTS> | <ROUTINE_CALL>

<**ROUTINE_DECLARATION**> ::= routine <IDEN> (<PARAMS>) {<PROG_STMT>} | “delay ” routine <IDEN> (<PARAMS>) {<PROG_STMT>}

<**PARAMS**> ::= IDEN* (, IDEN)*

<**IDEN**> ::= “_”* <ALPHABET>+ (<ALPHABET> | _ | <FIG_VALUES>)* | “_”+

<**VALUES**> ::= <FIG_VALUES> | <FUZZY_VALUES> | <SPECIAL_CHARS> | <YARN_VALUES> | <TEA_VALUES>

<**FIG_VALUES**> ::= <DIGIT>+ | “-” <NON_ZERO>$^{+}$

<**FUZZY_VALUES**> ::= <DIGIT>+ “.” <DIGIT>* | “-” <NON_ZERO>$^{+}$ “.” <DIGIT>*

<**YARN_VALUES**> ::= (“ “ | <YARN_CONTENT>)*

<**TEA_VALUES**> ::= “real” | “cap”

<**YARN_CONTENT**> ::= <ALPHABET> | <DIGIT> | <SPECIAL_CHARS>

<**ALPHABET**> ::= <LOW_KEY> | <HIGH_KEY>

<**LOW_KEY**> ::= “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”

<**HIGH_KEY**> ::= “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”

<**DIGIT**> ::= 0 | <NON_ZERO> (<NON_ZERO> | 0)*

<**NON_ZERO**> ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<**SPECIAL_CHARS**> ::= <OTHER_SYMBOLS> | <ARITH_SYMBOLS> | <DELIMITERS>

<OTHER_SYMBOLS> ::= “!” | “@” | “#” | “$” | “&” | “_” | “|” | “\” | “‘“ | “<“ | “>“ | “?” | “=”

<**ARITH_SYMBOL**> ::= “+” | “-” | “*” | “/” | “%” | “^”

<**UNARY_SYMBOL**> ::= “+” | “-” | “++” | “--” | “>>“ | “<<“

<**DELIMITERS**> ::= “,”| “.” | “(“ | “)” | “[“ | “]” | “{“ | “}”

<**ASSIGN_SYMBOL**> ::= “=” | “+=” | “-=” | “*=” | “/=” | “%=” | “^=”

<DOUBLE_QUOTE> ::= “””

DECLARATION STATEMENTS

<**LIT_DEC**> ::= “lit ” <IDEN> “;”| “lit ” <IDEN> “=” (<FIG_VALUES> | <FUZZY_VALUES> |
“””<YARN_VALUES>””” | <TEA_VALUES> | “ghosted”)”;”

<**FIRE_DEC**> ::= “fire ” <IDEN> “=” (<FIG_VALUES> | <FUZZY_VALUES>|
“””<YARN_VALUES>””” | <TEA_VALUES>)”;”

```
lit num
```

Leftmost Derivation

:: = <LIT_DEC> <IDEN>;
:: = lit <IDEN>;
:: = lit <ALPHABET>;
:: = lit <LOW_KEY>;
:: = lit n<ALPHABET>;
:: = lit n<LOW_KEY>;
:: = lit nu <ALPHABET>;
:: = lit nu<LOW_KEY>;
:: = lit num;

Rightmost Derivation

:: = <LIT_DEC> <IDEN>;
:: = < LIT_DEC > <ALPHABET>;
:: = < LIT_DEC > <LOW_KEY>;
:: = < LIT_DEC > <ALPHABET> m;
:: = < LIT_DEC > <LOW_KEY> m;
:: = < LIT_DEC > <ALPHABET> um;
:: = < LIT_DEC > <LOW_KEY> um;
:: = < LIT_DEC > <ALPHABET> num;
:: = < LIT_DEC > <LOW_KEY> num;
:: = < LIT_DEC > num;
:: = lit num;

```
fire Comp = 1
```

Left Derivation

::= <FIRE_DEC> <IDEN> <SPECIAL_CHARS> <FIG_VALUES>;
::= fire <ALPHABET> <SPECIAL_CHARS> <FIG_VALUES>;

::= fire <HIGH_KEY> <SPECIAL_CHARS> <FIG_VALUES>;
::= fire C <ALPHABET> <SPECIAL_CHARS> <FIG_VALUES>;
::= fire C <LOW_KEY> <SPECIAL_CHARS> <FIG_VALUES>;
::= fire Co <ALPHABET> <SPECIAL_CHARS> <FIG_VALUES>;
::= fire Co <LOW_KEY> <SPECIAL_CHARS> <FIG_VALUES>;
::= fire Com <ALPHABET> <SPECIAL_CHARS> <FIG_VALUES>;
::= fire Com <LOW_KEY> <SPECIAL_CHARS> <FIG_VALUES>;
::= fire Comp <SPECIAL_CHARS> <FIG_VALUES>;
::= fire Comp = <FIG_VALUES>;
::= fire Comp = <DIGIT>;
::= fire Comp = <NON_ZERO>;
::= fire Comp = 1;

Rightmost Derivation

::= <FIRE_DEC> <IDEN> <SPECIAL_CHARS> <FIG_VALUES>;
::= <FIRE_DEC> <IDEN> <SPECIAL_CHARS> <FIG_VALUES>;
::= <FIRE_DEC> <IDEN> <SPECIAL_CHARS> <DIGIT>;
::= <FIRE_DEC> <IDEN> <SPECIAL_CHARS> <NON_ZERO>;
::= <FIRE_DEC> <IDEN> <SPECIAL_CHARS> 1;
::= <FIRE_DEC> <ALPHABET> = 1;
::= <FIRE_DEC> <LOW_KEY> = 1;
::= <FIRE_DEC> <ALPHABET>p = 1;
::= <FIRE_DEC> <LOW_KEY>p = 1;
::= <FIRE_DEC> <ALPHABET>mp = 1;
::= <FIRE_DEC> <LOW_KEY>mp = 1;
::= <FIRE_DEC> <ALPHABET>omp = 1;
::= <FIRE_DEC> <HIGH_KEY>omp = 1;
::= <FIRE_DEC> Comp = 1;
::= fire Comp = 1;

`lit num01`

Leftmost Derivation

:: = <LIT_DEC> <IDEN>;
:: = lit <IDEN>;
::= lit <ALPHABET>;
::= lit <LOW_KEY>;
::= lit n <ALPHABET>;
::= lit n <LOW_KEY>;
::= lit nu <ALPHABET>;
::= lit nu <LOW_KEY>;
::= lit num <FIG_VALUES>;
::= lit num <DIGIT>;

::= lit num0 <FIG_VALUES>;
::= lit num0 <DIGIT>;
::= lit num0 <NON_ZERO>;
::= lit num01;

Rightmost Derivation

:: = <LIT_DEC> <IDEN>;
:: = <LIT_DEC> <IDEN>;
:: = <LIT_DEC> <FIG_VALUES>;
:: = <LIT_DEC> <DIGIT>;
:: = <LIT_DEC> <NON_ZERO>;
:: = <LIT_DEC> <FIG_VALUES>1;
:: = <LIT_DEC> <DIGIT>1;
:: = <LIT_DEC> <IDEN>01;
:: = <LIT_DEC> <ALPHABET>01;
:: = <LIT_DEC> <LOW_KEY>01;
:: = <LIT_DEC> <ALPHABET>m01;
:: = <LIT_DEC> <LOW_KEY>m01;
:: = <LIT_DEC> <ALPHABET>um01;
:: = <LIT_DEC> <LOW_KEY>um01;
:: = <LIT_DEC> num01;
:: = lit num01;

INPUT STATEMENTS

<**INPUT_STMTS**> ::= <IDEN> “= spill();” | “lit” <IDEN> “= spill();” | “fire”
<IDEN> “= spill();”

```
lit myVar = spill();
```

Leftmost Derivation

::= <INPUT_STMTS>;
::= "lit" <IDEN> "= spill();"
::= "lit" <ALPHABET> "= spill();"
::= "lit" <LOW_KEY> "= spill();"

::= "lit" m<LOW_KEY> "= spill();"
::= "lit" my<HIGH_KEY> "= spill();"
::= "lit" myV<LOW_KEY> "= spill();"
::= "lit" myVa<LOW_KEY> "= spill();"
::= "lit" myVar "= spill();"

Rightmost Derivation

::= <INPUT_STMTS>;
::= "lit" <IDEN> "= spill();"
::= "lit" <ALPHABET> "= spill();"
::= "lit" <LOW_KEY> "= spill();"
::= "lit" r<LOW_KEY> "= spill();"
::= "lit" ar<LOW_KEY> "= spill();"
::= "lit" Var<HIGH_KEY> "= spill();"
::= "lit" yVar<LOW_KEY> "= spill();"
::= "lit" myVar "= spill();"

```
fire myConst = spill();
```

Leftmost Derivation

::= <INPUT_STMTS>;
::= "lit" <IDEN> "= spill();"
::= "lit" <ALPHABET> "= spill();"
::= "lit" <LOW_KEY> "= spill();"
::= "lit" m<LOW_KEY> "= spill();"
::= "lit" my<HIGH_KEY> "= spill();"
::= "lit" myC<LOW_KEY> "= spill();"
::= "lit" myCo<LOW_KEY> "= spill();"
::= "lit" myCon<LOW_KEY> "= spill();"
::= "lit" myCons<LOW_KEY> "= spill();"
::= "lit" myConst "= spill();"

Rightmost Derivation

::= <INPUT_STMTS>;
::= "lit" <IDEN> "= spill();"
::= "lit" <ALPHABET> "= spill();"
::= "lit" <LOW_KEY> "= spill();"

::= "lit" t<LOW_KEY> "= spill();"
::= "lit" st<LOW_KEY> "= spill();"
::= "lit" nst<LOW_KEY> "= spill();"
::= "lit" onst<HIGH_KEY> "= spill();"
::= "lit" Const<LOW_KEY> "= spill();"
::= "lit" yConst<LOW_KEY> "= spill();"
::= "lit" myConst "= spill();"

```
varX = spill();
```

Leftmost Derivation

::= <INPUT_STMTS>;
::= "lit" <IDEN> "= spill();"
::= "lit" <ALPHABET> "= spill();"
::= "lit" <LOW_KEY> "= spill();"
::= "lit" v<LOW_KEY> "= spill();"
::= "lit" va<LOW_KEY> "= spill();"
::= "lit" var<HIGH_KEY> "= spill();"
::= "lit" varX "= spill();"

Rightmost Derivation

::= <INPUT_STMTS>;
::= "lit" <IDEN> "= spill();"
::= "lit" <ALPHABET> "= spill();"
::= "lit" <HIGH_KEY> "= spill();"
::= "lit" X<LOW_KEY> "= spill();"
::= "lit" rX<LOW_KEY> "= spill();"
::= "lit" arX<LOW_KEY> "= spill();"
::= "lit" varX "= spill();"

OUTPUT STATEMENTS

<**OUTPUT_STMTS**> ::= “flex” (<OUTPUT_LIST>)";"

<**OUTPUT_LIST**> ::= <OUTPUT_OBJECT> | <OUTPUT_LIST> "+" <OUTPUT_OBJECT>

<**OUTPUT_OBJECT**> ::= <IDEN> | <VALUES>

```
flex("hello world!");
```

Leftmost Derivation

::= <OUTPUT_STMTS>;
::= "flex"(<OUTPUT_LIST>) ";"
::= "flex"(<OUTPUT_OBJECT>) ";"
::= "flex"(<VALUES>)";"
::= "flex"(<YARN_VALUES>)";"
::= "flex"("<YARN_CONTENT>)";"
::= "flex"("<ALPHABET>)";"
::= "flex"("<LOW_KEY>)";"
::= "flex"("h<LOW_KEY>)";"
::= "flex"("he<LOW_KEY>)";"
::= "flex"("hel<LOW_KEY>)";"
::= "flex"("hell<LOW_KEY>)";"
::= "flex"("hello<LOW_KEY>)";"
::= "flex"("hello <YARN_VALUES>)";"
::= "flex"("hello <YARN_CONTENT>)";"
::= "flex"("hello <ALPHABET>)";"
::= "flex"("hello <LOW_KEY>)";"
::= "flex"("hello w<LOW_KEY>)";"
::= "flex"("hello wo<LOW_KEY>)";"
::= "flex"("hello wor<LOW_KEY>)";"
::= "flex"("hello worl<LOW_KEY>)";"
::= "flex"("hello world<SPECIAL_CHARS>)";"
::= "flex"("hello world!<YARN_VALUES>)";"
::= "flex"("hello world!")";"

Rightmost Derivation

::= <OUTPUT_STMTS>;

::= "flex"(<OUTPUT_LIST>) ";"
::= "flex"(<OUTPUT_OBJECT>) ";"
::= "flex"(<VALUES>)";"
::= "flex"(<YARN_VALUES>)";"
::= "flex"("<YARN_CONTENT>)";"
::= "flex"("<SPECIAL_CHARS>)";"
::= "flex"(!"<ALPHABET>)";"
::= "flex"(d!"<LOW_KEY>)";"
::= "flex"(ld!"<LOW_KEY>)";"
::= "flex"(old!"<LOW_KEY>)";"
::= "flex"(wold!"<YARN_VALUES>)";"
::= "flex"(wold!"<YARN_CONTENT>)";"
::= "flex"(wold!"<ALPHABET>)";"
::= "flex"(wold!"<LOW_KEY>)";"
::= "flex"(o wold!"<LOW_KEY>)";"
::= "flex"(lo wold!"<LOW_KEY>)";"
::= "flex"(llo wold!"<LOW_KEY>)";"
::= "flex"(ello wold!"<LOW_KEY>)";"
::= "flex"(hello wold!"<YARN_VALUES>)";"
::= "flex"("hello world!")";"

```
flex("hello" + IDENT);
```

Leftmost Derivation

::= <OUTPUT_STMTS>;
::= "flex"(<OUTPUT_LIST>) ";"
::= "flex"(<OUTPUT_OBJECT>) ";"
::= "flex"(<VALUES>)";"
::= "flex"(<YARN_VALUES>)";"
::= "flex"("<YARN_CONTENT>)";"
::= "flex"("<ALPHABET>)";"
::= "flex"("<LOW_KEY>)";"
::= "flex"("h<LOW_KEY>)";"
::= "flex"("he<LOW_KEY>)";"
::= "flex"("hel<LOW_KEY>)";"
::= "flex"("hell<LOW_KEY>)";"
::= "flex"("hello<YARN_VALUES>)";"
::= "flex"("hello" <OUTPUT_LIST>)";"

::= "flex"("hello" + <OUTPUT_OBJECT>)";"
::= "flex"("hello" + <IDEN>)";"
::= "flex"("hello" + <ALPHABET>)";"
::= "flex"("hello" + <HIGH_KEY>)";"
::= "flex"("hello" + I<HIGH_KEY>)";"
::= "flex"("hello" + ID<HIGH_KEY>)";"
::= "flex"("hello" + IDE<HIGH_KEY>)";"
::= "flex"("hello" + IDEN<HIGH_KEY>)";"
::= "flex"("hello" + IDENT)";"

Rightmost Derivation

::= <OUTPUT_STMTS>;
::= "flex"(<OUTPUT_LIST>)";"
::= "flex"(<OUTPUT_OBJECT>)";"
::= "flex"(<IDEN>)";"
::= "flex"(<ALPHABET>)";"
::= "flex"(<HIGH_KEY>)";"
::= "flex"(T<HIGH_KEY>)";"
::= "flex"(NT<HIGH_KEY>)";"
::= "flex"(ENT<HIGH_KEY>)";"
::= "flex"(DENT<HIGH_KEY>)";"
::= "flex"(IDENT<HIGH_KEY>)";"
::= "flex"(IDENT<OUTPUT_LIST>)";"
::= "flex"(+ IDENT<OUTPUT_OBJECT>)";"
::= "flex"(+ IDENT<VALUES>)";"
::= "flex"(+ IDENT<YARN_VALUES>)";"
::= "flex"(" + IDENT<YARN_CONTENT>)";"
::= "flex"(" + IDENT<ALPHABET>)";"
::= "flex"(" + IDENT<LOW_KEY>)";
::= "flex"(o" + IDENT<LOW_KEY>)";
::= "flex"(lo" + IDENT<LOW_KEY>)";
::= "flex"(llo" + IDENT<LOW_KEY>)";
::= "flex"(ello" + IDENT<LOW_KEY>)";
::= "flex"(hello" + IDENT<YARN_VALUES>)";
::= "flex"("hello" + IDENT)";

```
flex("STRING HERE" + expressionhere)
```

Leftmost Derivation

::= <OUTPUT_STMTS>;
::= "flex"(<OUTPUT_LIST>) ";"
::= "flex"(<OUTPUT_OBJECT>) ";"
::= "flex"(<VALUES>)";"
::= "flex"(<YARN_VALUES>)";"
::= "flex"("<YARN_CONTENT>)";"
::= "flex"("<ALPHABET>)";"
::= "flex"("<HIGH_KEY>)";"
::= "flex"("S<HIGH_KEY>)";"
::= "flex"("ST<HIGH_KEY>)";"
::= "flex"("STR<HIGH_KEY>)";"
::= "flex"("STRI<HIGH_KEY>)";"
::= "flex"("STRIN<HIGH_KEY>)";"
::= "flex"("STRING<YARN_VALUES>)";"
::= "flex"("STRING <HIGH_KEY>)";"
::= "flex"("STRING H<HIGH_KEY>)";"
::= "flex"("STRING HE<HIGH_KEY>)";"
::= "flex"("STRING HER<HIGH_KEY>)";"
::= "flex"("STRING HERE<YARN_VALUES>)";"
::= "flex"("STRING HERE"<OUTPUT_LIST>)";"
::= "flex"("STRING HERE" + <OUTPUT_OBJECT>)";"
::= "flex"("STRING HERE" + <IDEN>)";"
::= "flex"("STRING HERE" + <ALPHABET>)";"
::= "flex"("STRING HERE" + <LOW_KEY>)";"
::= "flex"("STRING HERE" + e<LOW_KEY>)";"
::= "flex"("STRING HERE" + ex<LOW_KEY>)";"
::= "flex"("STRING HERE" + exp<LOW_KEY>)";"
::= "flex"("STRING HERE" + exp<LOW_KEY>)";"
::= "flex"("STRING HERE" + expr<LOW_KEY>)";"
::= "flex"("STRING HERE" + expre<LOW_KEY>)";"
::= "flex"("STRING HERE" + expres<LOW_KEY>)";"
::= "flex"("STRING HERE" + express<LOW_KEY>)";"
::= "flex"("STRING HERE" + expressi<LOW_KEY>)";"
::= "flex"("STRING HERE" + expressio<LOW_KEY>)";"
::= "flex"("STRING HERE" + expressin<LOW_KEY>)";"
::= "flex"("STRING HERE" + expressinh<LOW_KEY>)";"
::= "flex"("STRING HERE" + expressinhe<LOW_KEY>)";"
::= "flex"("STRING HERE" + expressinher<LOW_KEY>)";"
::= "flex"("STRING HERE" + expressinhere)";"

Rightmost Derivation

::= <OUTPUT_STMTS>;
::= "flex"(<OUTPUT_LIST>)";"
::= "flex"(<OUTPUT_OBJECT>)";"
::= "flex"(<IDEN>)";"
::= "flex"(<ALPHABET>)";"
::= "flex"(<LOW_KEY>)";"
::= "flex"(e<LOW_KEY>)";"
::= "flex"(re<LOW_KEY>)";"
::= "flex"(ere<LOW_KEY>)";"
::= "flex"(here<LOW_KEY>)";"
::= "flex"(nhere<LOW_KEY>)";"
::= "flex"(onhere<LOW_KEY>)";"
::= "flex"(ionhere<LOW_KEY>)";"
::= "flex"(sionhere<LOW_KEY>)";"
::= "flex"(ssionhere<LOW_KEY>)";"
::= "flex"(essionhere<LOW_KEY>)";"
::= "flex"(ressionhere<LOW_KEY>)";"
::= "flex"(pressionhere<LOW_KEY>)";"
::= "flex"(xpressionhere<LOW_KEY>)";"
::= "flex"(expressionhere<OUTPUT_LIST>)";"
::= "flex"(+ expressionhere<OUTPUT_OBJECT>)";"
::= "flex"(+ expressionhere<VALUES>)";"
::= "flex"(+ expressionhere<YARN_VALUES>)";"
::= "flex"(" + expressionhere<YARN_CONTENT>)";"
::= "flex"(" + expressionhere<YARN_VALUES>)";"
::= "flex"(" + expressionhere<ALPHABET>)";"
::= "flex"(" + expressionhere<HIGH_KEY>)";"
::= "flex"(E" + expressionhere<HIGH_KEY>)";"
::= "flex"(RE" + expressionhere<HIGH_KEY>)";"
::= "flex"(ERE" + expressionhere<HIGH_KEY>)";"
::= "flex"(HERE" + expressionhere<YARN_VALUES>)";"
::= "flex"(HERE" + expressionhere<HIGH_KEY>)";"
::= "flex"(G HERE" + expressionhere<HIGH_KEY>)";"
::= "flex"(NG HERE" + expressionhere<HIGH_KEY>)";"
::= "flex"(ING HERE" + expressionhere<HIGH_KEY>)";"
::= "flex"(RING HERE" + expressionhere<HIGH_KEY>)";"
::= "flex"(TRING HERE" + expressionhere<HIGH_KEY>)";"
::= "flex"(STRING HERE" + expressionhere<HIGH_KEY>)";"

::= "flex"(STRING HERE" + expressionhere<YARN_VALUES>)";"
::= "flex"("STRING HERE" + expressionhere)";"

ASSIGNMENT STATEMENTS

<**ASSIGN_STMTS** > ::= <IDEN> <ASSIGN_SYMBOL> <VALUES> ";"

```
t = 9;
```

Leftmost Derivation

::= <ASSIGN_STMTS >;
::= <IDEN> <ASSIGN_SYMBOL> <VALUES>;
::= <ALPHABET> <ASSIGN_SYMBOL> <VALUES>;
::= <LOW_KEY> <ASSIGN_SYMBOL> <VALUES>;
::= t <ASSIGN_SYMBOL> <VALUES>;
::= t = <VALUES>;
::= t = <FIG_VALUES>;
::= t = <DIGIT>;
::= t = <NON_ZERO>;
::= t = 9;

Rightmost Derivation

::= <ASSIGN_STMTS >;
::= <IDEN> <ASSIGN_SYMBOL> <VALUES>;
::= <IDEN> <ASSIGN_SYMBOL> <FIG_VALUES>;
::= <IDEN> <ASSIGN_SYMBOL> <DIGIT>;
::= <IDEN> <ASSIGN_SYMBOL> <NON_ZERO>;
::= <IDEN> <ASSIGN_SYMBOL> 9;
::= <IDEN> = 9;
::= <IDEN> = 9;
::= <ALPHABET> = 9;

::= <LOW_KEY> = 9;
::= t = 9;

kpop = "TWICE";

Leftmost Derivation

::= <ASSIGN_STMTS >;
::= <IDEN> <ASSIGN_SYMBOL> <VALUES>;
::= <ALPHABET> <ASSIGN_SYMBOL> <VALUES>;
::= <LOW_KEY> <ASSIGN_SYMBOL> <VALUES>;
::= k<LOW_KEY> <ASSIGN_SYMBOL> <VALUES>;
::= kp<LOW_KEY> <ASSIGN_SYMBOL> <VALUES>;
::= kpo<LOW_KEY> <ASSIGN_SYMBOL> <VALUES>;
::= kpop <ASSIGN_SYMBOL> <VALUES>;
::= kpop = <VALUES>;
::= kpop = <SPECIAL_CHARS>;
::= kpop = "<CHAR_VALUE>;
::= kpop = "<ALPHABET>;
::= kpop = "<HIGH_KEY>;
::= kpop = "T<HIGH_KEY>;
::= kpop = "TW<HIGH_KEY>;
::= kpop = "TWI<HIGH_KEY>;
::= kpop = "TWIC<HIGH_KEY>;
::= kpop = "TWICE<SPECIAL_CHAR>;
::= kpop = "TWICE";

Rightmost Derivation

::= <ASSIGN_STMTS >;
::= <IDEN> <ASSIGN_SYMBOL> <VALUES>;
::= <IDEN> <ASSIGN_SYMBOL> <SPECIAL_CHAR>;
::= <IDEN> <ASSIGN_SYMBOL> <ALPHABET>";
::= <IDEN> <ASSIGN_SYMBOL> <HIGH_KEY>";
::= <IDEN> <ASSIGN_SYMBOL> <HIGH_KEY>E";
::= <IDEN> <ASSIGN_SYMBOL> <HIGH_KEY>CE";
::= <IDEN> <ASSIGN_SYMBOL> <HIGH_KEY>ICE";
::= <IDEN> <ASSIGN_SYMBOL> <HIGH_KEY>WICE";
::= <IDEN> <ASSIGN_SYMBOL> <SPECIAL_CHAR>TWICE";
::= <IDEN> <ASSIGN_SYMBOL> "TWICE";

::= <IDEN> = “TWICE”;
::= <ALPHABET> = “TWICE”;
::= <LOW_KEY> = “TWICE”;
::= <LOW_KEY>p = “TWICE”;
::= <LOW_KEY>op = “TWICE”;
::= <LOW_KEY>pop = “TWICE”;
::= kpop = “TWICE”;

```
money += 100;
```

Leftmost Derivation

::= <ASSIGN_STMTS >;
::= <IDEN> <ASSIGN_SYMBOL> <VALUES>;
::= <ALPHABET> <ASSIGN_SYMBOL> <VALUES>;
::= <LOW_KEY> <ASSIGN_SYMBOL> <VALUES>;
::= m<LOW_KEY> <ASSIGN_SYMBOL> <VALUES>;
::= mo<LOW_KEY> <ASSIGN_SYMBOL> <VALUES>;
::= mon<LOW_KEY> <ASSIGN_SYMBOL> <VALUES>;
::= mone<LOW_KEY> <ASSIGN_SYMBOL> <VALUES>;
::= money <ASSIGN_SYMBOL> <VALUES>;
::= money += <VALUES>;
::= money += <FIG_VALUES>;
::= money += <DIGIT>;
::= money += <NO_ZER0>;
::= money += 1<VALUES>;
::= money += 1<FIG_VALUES>;
::= money += 1<DIGIT>;
::= money += 10<DIGIT>;
::= money += 100;

Rightmost Derivation

::= <ASSIGN_STMTS >;
::= <IDEN> <ASSIGN_SYMBOL> <VALUES>;
::= <IDEN> <ASSIGN_SYMBOL> <FIG_VALUES>;
::= <IDEN> <ASSIGN_SYMBOL> <DIGIT>;
::= <IDEN> <ASSIGN_SYMBOL> <DIGIT>0;
::= <IDEN> <ASSIGN_SYMBOL> <VALUES>00;
::= <IDEN> <ASSIGN_SYMBOL> <FIG_VALUES>00;

::= <IDEN> <ASSIGN_SYMBOL> <DIGIT>00;
::= <IDEN> <ASSIGN_SYMBOL> <NON_ZERO>00;
::= <IDEN> <ASSIGN_SYMBOL> 100;
::= <IDEN> += 100;
::= <ALPHABET> += 100;
::= <LOW_KEY> += 100;
::= <LOW_KEY>y += 100;
::= <LOW_KEY>ey += 100;
::= <LOW_KEY>ney += 100;
::= <LOW_KEY>oney += 100;
::= money += 100;

CONDITIONAL STATEMENTS

<**COND_STMT**> ::= <YEET> <YIKES>* | <YEET> <YIKES>* <YAS>

<**YEET**> ::= “yeet (” <CONDITION> “){“ <PROG_STMT> “}”

<**YIKES**> ::= “yikes (” <CONDITION> “){“ <PROG_STMT> “}”

<**YAS**> ::= “yas {” <PROG_STMT> “}”

<**CONDITION**> ::= <REL_EXPR> | <COND_LOGIC_EXPR>

<**REL_EXPR**> ::= (<BOOL_EXPR> | <IDEN> | <VALUES>) <REL_SYMBOL> (IDEN | VALUES) | <REL_EXPR> (<REL_SYMBOL> <REL_EXPR>)*

<**REL_SYMBOL**> ::= “<” | “>” | “==” | “<=” | “>=” | “!=”

<LOG_EXPR> ::= (<VALUE> | <REL_EXPR>) <LOG_SYMBOL> (<VALUE> | <REL_EXPR>) | <LOG_EXPR> (<LOG_SYMBOL> (<VALUE> | <REL_EXPR>))* | <NOT_EXPR>

<**LOG_SYMBOL**> ::= “||” | “&&”

<NOT_EXPR> ::= “!“ (<VALUE> | <CONDITION>)

```
yeet(n==0){flex(“n is zero”);}
```

Leftmost Derivation

::= <COND_STMT>
::= <YEET>
::= “yeet (” <CONDITION> “){“ <PROG_STMT> “}”
::= “yeet (” <REL_EXPR> “){“ <PROG_STMT> “}”
::= “yeet (” <IDEN> <REL_SYMBOLS> <VALUE> “){“ <PROG_STMT> “}”
::= “yeet (” n <REL_SYMBOLS> <VALUE> “){“ <PROG_STMT> “}”
::= “yeet (” n == <VALUE> “){“ <PROG_STMT> “}”
::= “yeet (” n == <FIG_VALUES> “){“ <PROG_STMT> “}”
::= “yeet (” n == <DIGIT> “){“ <PROG_STMT> “}”
::= “yeet (” n == 0 “){“ <PROG_STMT> “}”
::= “yeet (” n == 0 “){ flex(”<OUTPUT_LIST>“) ; }”
::= “yeet (” n == 0 “){ flex(”<OUTPUT_OBJECTS>“) ; }”
::= “yeet (” n == 0 “){ flex(“”<VALUES>“”) ; }”
::= “yeet (” n == 0 “){ flex(“”<YARN_VALUES>“”) ; }”
::= “yeet (” n == 0 “){ flex(“”<YARN_CONTENTS>“”) ; }”
::= “yeet (” n == 0 “){ flex(“”<ALPHABET>“”) ; }”
::= “yeet (” n == 0 “){ flex(“”<LOW_KEY>“”) ; }”
::= “yeet (” n == 0 “){ flex(“”n<LOW_KEY>“”) ; }”
::= “yeet (” n == 0 “){ flex(“”n<YARN_VALUES>“”) ; }”
::= “yeet (” n == 0 “){ flex(“”n <LOW_KEY>“”) ; }”
::= “yeet (” n == 0 “){ flex(“”n i<LOW_KEY>“”) ; }”
::= “yeet (” n == 0 “){ flex(“”n is<LOW_KEY>“”) ; }”
::= “yeet (” n == 0 “){ flex(“”n is<YARN_VALUES>“”) ; }”
::= “yeet (” n == 0 “){ flex(“”n is <LOW_KEY>“”) ; }”
::= “yeet (” n == 0 “){ flex(“”n is z<LOW_KEY>“”) ; }”
::= “yeet (” n == 0 “){ flex(“”n is ze<LOW_KEY>“”) ; }”
::= “yeet (” n == 0 “){ flex(“”n is zer<LOW_KEY>) ″; }”
::= “yeet (” n == 0 “){ flex(“”n is zero“”) ; }”

Rightmost Derivation

::= <COND_STMT>

::= <YEET>
::= “yeet (” <CONDITION> “){“ <PROG_STMT> “}”
::= “yeet (” <CONDITION> “){ flex(“”<OUTPUT_LIST>“”) ; }”
::= “yeet (” <CONDITION> “){ flex(“”<OUTPUT_OBJECTS>“”) ; }”
::= “yeet (” <CONDITION> “){ flex(” <VALUES> “) ; }”
::= “yeet (” <CONDITION> “){ flex(“”YARN_VALUES>“”) ; }”
::= “yeet (” <CONDITION> “){ flex(“”<YARN_CONTENTS>“”) ; }”
::= “yeet (” <CONDITION> “){ flex(“”<ALPHABET>“”) ; }”
::= “yeet (” <CONDITION> “){ flex(“”<LOW_KEY>“”) ; }”
::= “yeet (” <CONDITION> “){ flex(“”<LOW_KEY>o“”) ; }”
::= “yeet (” <CONDITION> “){ flex(“”<LOW_KEY>ro“”) ; }”
::= “yeet (” <CONDITION> “){ flex(“”<LOW_KEY>ero“”) ; }”
::= “yeet (” <CONDITION> “){ flex(“”<YARN_VALUES>zero“”) ; }”
::= “yeet (” <CONDITION> “){ flex(“”<LOW_KEY> zero“”) ; }”
::= “yeet (” <CONDITION> “){ flex(“”<LOW_KEY>s zero“”) ; }”
::= “yeet (” <CONDITION> “){ flex(“”<YARN_VALUES>is zero“”) ; }”
::= “yeet (” <CONDITION> “){ flex(“”<LOW_KEY> is zero“”) ; }”
::= “yeet (” <CONDITION> “){ flex(“”n is zero“”) ; }”
::= “yeet (” <REL_EXPR> “){ flex(“”n is zero“”) ; }”
::= “yeet (” <IDEN> <REL_SYMBOLS> <VALUE> “){ flex(“”n is zero“”) ; }”
::= “yeet (” <IDEN> <REL_SYMBOLS> <FIG_VALUE> “){ flex(“”n is zero“”) ; }”
::= “yeet (” <IDEN> <REL_SYMBOLS> <DIGIT> “){ flex(“”n is zero“”) ; }”
::= “yeet (” <IDEN> <REL_SYMBOLS> 0 “){ flex(“”n is zero“”) ; }”
::= “yeet (” <IDEN> == 0 “){ flex(“”n is zero“”) ; }”
::= “yeet (” n == 0 “){ flex(“”n is zero“”) ; }

```
yeet(n>0){flex(“positive”);}
yikes(n<0){flex(“negative”);}
```

Leftmost Derivation

::= <COND_STMT>
::= <YEET> <YIKES>
::= “yeet (” <CONDITION> “){“ <PROG_STMT> “}” <YIKES>
::= “yeet (” <REL_EXPR> “){“ <PROG_STMT> “}” <YIKES>
::= “yeet (” <IDEN> <REL_SYMBOLS> <VALUE> “){“ <PROG_STMT> “}” <YIKES>
::= “yeet (” n <REL_SYMBOLS> <VALUE> “){“ <PROG_STMT> “}” <YIKES>
::= “yeet (” n > <VALUE> “){“ <PROG_STMT> “}” <YIKES>

::= “yeet (” n > <FIG_VALUES> “){“ <PROG_STMT> “}” <YIKES>
::= “yeet (” n > <DIGIT> “){“ <PROG_STMT> “}” <YIKES>
::= “yeet (” n > 0 “){“ <PROG_STMT> “}” <YIKES>
::= “yeet (” n > 0 “){ flex(“”<OUTPUT_LIST>“”) ; }” <YIKES>
::= “yeet (” n > 0 “){ flex(“”<OUTPUT_OBJECTS>“”) ; }” <YIKES>
::= “yeet (” n > 0 “){ flex(“”<VALUES>“”) ; }” <YIKES>
::= “yeet (” n > 0 “){ flex(“”YARN_VALUES>“”) ; }” <YIKES>
::= “yeet (” n > 0 “){ flex(“”<YARN_CONTENTS>“”) ; }” <YIKES>
::= “yeet (” n > 0 “){ flex(“”<ALPHABET>“”) ; }” <YIKES>
::= “yeet (” n > 0 “){ flex(“”<LOW_KEY>“”) ; }” <YIKES>
::= “yeet (” n > 0 “){ flex(“”p<LOW_KEY>“”) ; }” <YIKES>
::= “yeet (” n > 0 “){ flex(“”po<LOW_KEY>“”) ; }” <YIKES>
::= “yeet (” n > 0 “){ flex(“”pos<LOW_KEY>“”) ; }” <YIKES>
::= “yeet (” n > 0 “){ flex(“”posi<LOW_KEY>“”) ; }”” <YIKES>
::= “yeet (” n > 0 “){ flex(“”posit<LOW_KEY>“”) ; }” <YIKES>
::= “yeet (” n > 0 “){ flex(“”positi<LOW_KEY>“”) ; }” <YIKES>
::= “yeet (” n > 0 “){ flex(“”positiv<LOW_KEY>“”) ; }” <YIKES>
::= “yeet (” n > 0 “){ flex(“”positive“”) ; }” “yikes (” <REL_EXPR> “){“ <PROG_STMT> “}”
::= “yeet (” n > 0 “){ flex(“”positive“”) ; }” “yikes (” <IDEN> <REL_SYMBOLS> <VALUE>
“){“ <PROG_STMT> “}”
::= “yeet (” n > 0 “){ flex(“”positive“”) ; }” “yikes (” n <REL_SYMBOLS> <VALUE> “){“
<PROG_STMT> “}”
::= “yeet (” n > 0 “){ flex(“”positive“”) ; }” “yikes (” n < <VALUE> “){“ <PROG_STMT> “}”
::= “yeet (” n > 0 “){ flex(“”positive“”) ; }” “yikes (” n < <FIG_VALUE> “){“ <PROG_STMT>
“}”
::= “yeet (” n > 0 “){ flex(“”positive“”) ; }” “yikes (” n < <DIGIT> “){“ <PROG_STMT> “}”
::= “yeet (” n > 0 “){ flex(“”positive“”) ; }” “yikes (” n < 0 “){“ <PROG_STMT> “}”
::= “yeet (” n > 0 “){ flex(“”positive“”) ; }” “yikes (” n < 0 “){ flex(“”<OUTPUT_LIST>“”) ; }”
::= “yeet (” n > 0 “){ flex(“”positive“”) ; }” “yikes (” n < 0 “){
flex(“”<OUTPUT_OBJECTS>“”) ; }”
::= “yeet (” n > 0 “){ flex(“”positive“”) ; }” “yikes (” n < 0 “){ flex(“”<VALUES>“”) ; }”
::= “yeet (” n > 0 “){ flex(“”positive“”) ; }” “yikes (” n < 0 “){ flex(“”<YARN_VALUES>“”) ;
}”
::= “yeet (” n > 0 “){ flex(“”positive“”) ; }” “yikes (” n < 0 “){
flex(“”<YARN_CONTENTS>“”) ; }”
::= “yeet (” n > 0 “){ flex(“”positive“”) ; }” “yikes (” n < 0 “){ flex(“”<ALPHABET>“”) ; }”
::= “yeet (” n > 0 “){ flex(“”positive“”) ; }” “yikes (” n < 0 “){ flex(“”<LOW_KEY>“”) ; }”
::= “yeet (” n > 0 “){ flex(“”positive“”) ; }” “yikes (” n < 0 “){ flex(“”n<LOW_KEY>“”) ; }”
::= “yeet (” n > 0 “){ flex(“”positive“”) ; }” “yikes (” n < 0 “){ flex(“”ne<LOW_KEY>“”) ; }”
::= “yeet (” n > 0 “){ flex(“”positive“”) ; }” “yikes (” n < 0 “){ flex(“”neg<LOW_KEY>“”) ; }”

::= “yeet (” n > 0 “){ flex(“”positive“”) ; }” “yikes (” n < 0 “){ flex(“”nega<LOW_KEY>“”) ; }”

::= “yeet (” n > 0 “){ flex(“”positive“”) ; }” “yikes (” n < 0 “){ flex(“”negat<LOW_KEY>“”) ; }”

::= “yeet (” n > 0 “){ flex(“”positive“”) ; }” “yikes (” n < 0 “){ flex(“”negati<LOW_KEY>“”) ; }”

::= “yeet (” n > 0 “){ flex(“”positive“”) ; }” “yikes (” n < 0 “){ flex(“”negativ<LOW_KEY>“”) ; }”

::= “yeet (” n > 0 “){ flex(“”positive“”) ; }” “yikes (” n < 0 “){ flex(“”negative“”) ; }”

Rightmost Derivation

::= <COND_STMT>

::= <YEET> <YIKES>

::= <YEET> “yikes (” <CONDITION> “){“ <PROG_STMT> “}”

::= <YEET> “yikes (” <CONDITION> “){“ <OUTPUT_LIST> “}”

::= <YEET> “yikes (” <CONDITION> “){“ <OUTPUT_OBJECTS> “}”

::= <YEET> “yikes (” <CONDITION> “){“” <VALUES> “”}”

::= <YEET> “yikes (” <CONDITION> “){“” <YARN_VALUES> “”}”

::= <YEET> “yikes (” <CONDITION> “){“” <YARN_CONTENTS> “”}”

::= <YEET> “yikes (” <CONDITION> “){“” <ALPHABET> “”}”

::= <YEET> “yikes (” <CONDITION> “){“” <LOW_KEY> “”}”

::= <YEET> “yikes (” <CONDITION> “){“ flex(“” <LOW_KEY>e “”) ; }”

::= <YEET> “yikes (” <CONDITION> “){“ flex(“” <LOW_KEY>ve “”) ; }”

::= <YEET> “yikes (” <CONDITION> “){“ flex(“” <LOW_KEY>ive “”) ; }”

::= <YEET> “yikes (” <CONDITION> “){“ flex(“” <LOW_KEY>tive “”) ; }”

::= <YEET> “yikes (” <CONDITION> “){“ flex(“” <LOW_KEY>ative “”) ; }”

::= <YEET> “yikes (” <CONDITION> “){“ flex(“” <LOW_KEY>gative “”) ; }”

::= <YEET> “yikes (” <CONDITION> “){“ flex(“” <LOW_KEY>egative “”) ; }”

::= <YEET> “yikes (” <CONDITION> “){“ flex(“”negative“”) ; }”

::= <YEET> “yikes (” <REL_EXPR> “){“ flex(“”negative“”) ; }”

::= <YEET> “yikes (” <IDEN> <REL_SYMBOLS> <VALUE> “){“ flex(“”negative“”) ; }”

::= <YEET> “yikes (” <IDEN> <REL_SYMBOLS> <FIG_VALUE> “){“ flex(“”negative“”) ; }”

::= <YEET> “yikes (” <IDEN> <REL_SYMBOLS> <DIGIT> “){“ flex(“”negative“”) ; }”

::= <YEET> “yikes (” <IDEN> <REL_SYMBOLS> 0 “){“ flex(“”negative“”) ; }”

::= <YEET> “yikes (” <IDEN> < 0 “){“ flex(“”negative“”) ; }”

::= <YEET> “yikes (” n < 0 “){“ flex(“”negative“”) ; }”

::= “yeet (” <CONDITION> “){“ <PROG_STMT> “} yikes (” n < 0 “){“ flex(“”negative“”) ; }”

::= “yeet (” <CONDITION> “){ flex(” <OUTPUT_LIST> “) ; } yikes (” n < 0 “){“ flex(“”negative“”) ; }”

::= “yeet (” <CONDITION> “){ flex(” <OUTPUT_OBJECTS> “) ; } yikes (” n < 0 “){“
flex(“”negative“”) ; }”
::= “yeet (” <CONDITION> “){ flex(” <VALUES> “) ; } yikes (” n < 0 “){“ flex(“”negative“”) ;
}”
::= “yeet (” <CONDITION> “){ flex(“”<YARN_VALUES>“”) ; } yikes (” n < 0 “){“
flex(“”negative“”) ; }”
::= “yeet (” <CONDITION> “){ flex(“”<YARN_CONTENTS>“”) ; } yikes (” n < 0 “){“
flex(“”negative“”) ; }”
::= “yeet (” <CONDITION> “){ flex(“”<ALPHABET>“”) ; } yikes (” n < 0 “){“
flex(“”negative“”) ; }”
::= “yeet (” <CONDITION> “){ flex(“”<LOW_KEY>“”) ; } yikes (” n < 0 “){“
flex(“”negative“”) ; }”
::= “yeet (” <CONDITION> “){ flex(“”<LOW_KEY>e“”) ; } yikes (” n < 0 “){“
flex(“”negative“”) ; }”
::= “yeet (” <CONDITION> “){ flex(“”<LOW_KEY>ve“”) ; } yikes (” n < 0 “){“
flex(“”negative“”) ; }”
::= “yeet (” <CONDITION> “){ flex(“”<LOW_KEY>ive“”) ; } yikes (” n < 0 “){“
flex(“”negative“”) ; }”
::= “yeet (” <CONDITION> “){ flex(“”<LOW_KEY>tive“”) ; } yikes (” n < 0 “){“
flex(“”negative“”) ; }”
::= “yeet (” <CONDITION> “){ flex(“”<LOW_KEY>itive“”) ; } yikes (” n < 0 “){“
flex(“”negative“”) ; }”
::= “yeet (” <CONDITION> “){ flex(“”<LOW_KEY>sitive“”) ; } yikes (” n < 0 “){“
flex(“”negative“”) ; }”
::= “yeet (” <CONDITION> “){ flex(“”<LOW_KEY>ositive“”) ; } yikes (” n < 0 “){“
flex(“”negative“”) ; }”
::= “yeet (” <CONDITION> “){ flex(“”positive“”) ; } yikes (” n < 0 “){“ flex(“”negative“”) ;
}”
::= “yeet (” <REL_EXPR> “){ flex(“”positive“”) ; } yikes (” n < 0 “){“ flex(“”negative“”) ; }”
::= “yeet (” <IDEN> <REL_SYMBOLS> <VALUE> “){ flex(“”positive“”) ; } yikes (” n < 0 “){“
flex(“”negative“”) ; }”
::= “yeet (” <IDEN> <REL_SYMBOLS> <FIG_VALUE> “){ flex(“”positive“”) ; } yikes (” n < 0
“){“ flex(“”negative“”) ; }”
::= “yeet (” <IDEN> <REL_SYMBOLS> <DIGIT> “){ flex(“”positive“”) ; } yikes (” n < 0 “){“
flex(“”negative“”) ; }”
::= “yeet (” <IDEN> <REL_SYMBOLS> 0 “){ flex(“”positive“”) ; } yikes (” n < 0 “){“
flex(“”negative“”) ; }
::= “yeet (” <IDEN> >0 “){ flex(“”positive“”) ; } yikes (” n < 0 “){“ flex(“”negative“”) ; }
::= “yeet (” n >0 “){ flex(“”positive“”) ; } yikes (” n < 0 “){“ flex(“”negative“”) ; }

```
yeet(n>0 && n<0){flex("nonzero");}
```

Leftmost Derivation

::= <COND_STMT>
::= <YEET>
::= “yeet (” <CONDITION> “){“ <PROG_STMT> “}”
::= “yeet (” <LOG_EXPR> “){“ <PROG_STMT> “}”
::= “yeet (” <REL_EXPR> <LOG_SYMBOL> <REL_EXPR> “){“ <PROG_STMT> “}”
::= “yeet (” <IDEN> <REL_SYMBOLS> <VALUE> <LOG_SYMBOL> <REL_EXPR> “){“ <PROG_STMT> “}”
::= “yeet (” n <REL_SYMBOLS> <VALUE> <LOG_SYMBOL> <REL_EXPR> “){“ <PROG_STMT> “}”
::= “yeet (” n > <VALUE> <LOG_SYMBOL> <REL_EXPR> “){“ <PROG_STMT> “}”
::= “yeet (” n > <FIG_VALUE> <LOG_SYMBOL> <REL_EXPR> “){“ <PROG_STMT> “}”
::= “yeet (” n > <DIGIT> <LOG_SYMBOL> <REL_EXPR> “){“ <PROG_STMT> “}”
::= “yeet (” n > 0 <LOG_SYMBOL> <REL_EXPR> “){“ <PROG_STMT> “}”
::= “yeet (” n > 0 && <REL_EXPR> “){“ <PROG_STMT> “}”
::= “yeet (” n > 0 && n <REL_SYMBOLS> <VALUE> “){“ <PROG_STMT> “}”
::= “yeet (” n > 0 && n < <VALUE> “){“ <PROG_STMT> “}”
::= “yeet (” n > 0 && n < <FIG_VALUE> “){“ <PROG_STMT> “}”
::= “yeet (” n > 0 && n < DIGIT> “){“ <PROG_STMT> “}”
::= “yeet (” n>0 && n<0 “){“ <PROG_STMT> “}”
::= “yeet (” n>0 && n<0 “){ flex(”<OUTPUT_LIST>“) ; }”
::= “yeet (” n>0 && n<0 “){ flex(”<OUTPUT_OBJECTS>“) ; }”
::= “yeet (” n>0 && n<0 “){ flex(“”<VALUES>“”) ; }”
::= “yeet (” n>0 && n<0 “){ flex(“”<YARN_VALUES>“”) ; }”
::= “yeet (” n>0 && n<0 “){ flex(“”<YARN_CONTENTS>“”) ; }”
::= “yeet (” n>0 && n<0 “){ flex(“”<ALPHABET>“”) ; }”
::= “yeet (” n>0 && n<0 “){ flex(“”<LOW_KEY>“”) ; }”
::= “yeet (” n>0 && n<0 “){ flex(“”n<LOW_KEY>“”) ; }”
::= “yeet (” n>0 && n<0 “){ flex(“”no<LOW_KEY>“”) ; }”
::= “yeet (” n>0 && n<0 “){ flex(“”non<LOW_KEY>“”) ; }”
::= “yeet (” n>0 && n<0 “){ flex(“”nonz<YARN_VALUES>“”) ; }”
::= “yeet (” n>0 && n<0 “){ flex(“”nonze<LOW_KEY>“”) ; }”
::= “yeet (” n>0 && n<0 “){ flex(“”nonzer<LOW_KEY>“”) ; }”
::= “yeet (” n>0 && n<0 “){ flex(“”nonzero<LOW_KEY>“”) ; }”

Rightmost Derivation

::= <COND_STMT>
::= <YEET>
::= “yeet (” <CONDITION> “){“ <PROG_STMT> “}”
::= “yeet (” <CONDITION> “){ flex(“”<OUTPUT_LIST>“”) ; }”
::= “yeet (” <CONDITION> “){ flex(“”<OUTPUT_OBJECTS>“”) ; }”
::= “yeet (” <CONDITION> “){ flex(” <VALUES> “) ; }”
::= “yeet (” <CONDITION> “){ flex(“”YARN_VALUES>“”) ; }”
::= “yeet (” <CONDITION> “){ flex(“”<YARN_CONTENTS>“”) ; }”
::= “yeet (” <CONDITION> “){ flex(“”<ALPHABET>“”) ; }”
::= “yeet (” <CONDITION> “){ flex(“”<LOW_KEY>“”) ; }”
::= “yeet (” <CONDITION> “){ flex(“”<LOW_KEY>o“”) ; }”
::= “yeet (” <CONDITION> “){ flex(“”<LOW_KEY>ro“”) ; }”
::= “yeet (” <CONDITION> “){ flex(“”<LOW_KEY>ero“”) ; }”
::= “yeet (” <CONDITION> “){ flex(“”<YARN_VALUES>zero“”) ; }”
::= “yeet (” <CONDITION> “){ flex(“”<YARN_VALUES>nzero“”) ; }”
::= “yeet (” <CONDITION> “){ flex(“”<YARN_VALUES>onzero“”) ; }”
::= “yeet (” <CONDITION> “){ flex(“”nonzero“”) ; }”
:= “yeet (” <LOG_EXPR> “){ flex(“”nonzero“”) ; }”
:= “yeet (” <REL_EXPR> <LOG_SYMBOL> <REL_EXPR> “){ flex(“”nonzero“”) ; }”
:= “yeet (” <REL_EXPR> <LOG_SYMBOL> <IDEN> <REL_SYMBOLS> <VALUE> “){
flex(“”nonzero“”) ; }”
:= “yeet (” <REL_EXPR> <LOG_SYMBOL> <IDEN> <REL_SYMBOLS> <FIG_VALUE> “){
flex(“”nonzero“”) ; }”
:= “yeet (” <REL_EXPR> <LOG_SYMBOL> <IDEN> <REL_SYMBOLS> <DIGIT> “){
flex(“”nonzero“”) ; }”
:= “yeet (” <REL_EXPR> <LOG_SYMBOL> <IDEN> <REL_SYMBOLS> 0 “){
flex(“”nonzero“”) ; }”
:= “yeet (” <REL_EXPR> <LOG_SYMBOL> <IDEN> < 0 “){ flex(“”nonzero“”) ; }”
:= “yeet (” <REL_EXPR> <LOG_SYMBOL> n < 0 “){ flex(“”nonzero“”) ; }”
:= “yeet (” <REL_EXPR> && n < 0 “){ flex(“”nonzero“”) ; }”
:= “yeet (” <IDEN> <REL_SYMBOLS> <VALUE> && n < 0 “){ flex(“”nonzero“”) ; }”
:= “yeet (” <IDEN> <REL_SYMBOLS> <FIG_VALUE> && n < 0 “){ flex(“”nonzero“”) ; }”
:= “yeet (” <IDEN> <REL_SYMBOLS> <DIGIT> && n < 0 “){ flex(“”nonzero“”) ; }”
:= “yeet (” <IDEN> <REL_SYMBOLS> 0 && n < 0 “){ flex(“”nonzero“”) ; }”
:= “yeet (” <IDEN> > 0 && n < 0 “){ flex(“”nonzero“”) ; }”
::= “yeet (” n>0 && n<0 “){ flex(“”n is zero“”) ; }

STEPWISE INCREMENT AND DECREMENT

<**STEPWISE_STATEMENT**> ::= <INCREMENT_STATEMENT> |
<DECREMENT_STATEMENT>

<**INCREMENT_STATEMENT**> ::= <VAR> ">>" <FIG_VALUES> | <EXPRESSION> ";"

<**DECREMENT_STATEMENT**> ::= <VAR> "<<" <FIG_VALUES> | <EXPRESSION> ";"

```
incrementNum >> 42;
```

Leftmost Derivation

::= <STEPWISE_STATEMENT>
::= <INCREMENT_STATEMENT>;
::= <VAR> >> <FIG_VALUES>;
::= <IDEN> >> <FIG_VALUES>;
::= <ALPHABET> >> <FIG_VALUES>;
::= <LOW_KEY> >> <FIG_VALUES>;
::= i< ALPHABET > >> <FIG_VALUES>;
::= i<LOW_KEY> >> <FIG_VALUES>;
::= in<ALPHABET> >> <FIG_VALUES>;
::= in<LOW_KEY> >> <FIG_VALUES>;
::= inc<ALPHABET> >> <FIG_VALUES>;
::= inc< LOW_KEY > >> <FIG_VALUES>;
::= incr<ALPHABET> >> <FIG_VALUES>;
::= incr<LOW_KEY> >> <FIG_VALUES>;
::= incre<ALPHABET> >> <FIG_VALUES>;
::= incre<LOW_KEY> >> <FIG_VALUES>;
::= increm<ALPHABET> >> <FIG_VALUES>;
::= increm<LOW_KEY> >> <FIG_VALUES>;
::= increme<ALPHABET> >> <FIG_VALUES>;
::= increme<LOW_KEY> >> <FIG_VALUES>;
::= incremen<ALPHABET> >> <FIG_VALUES>;

::= incremen<LOW_KEY> >> <FIG_VALUES>;
::= increment<ALPHABET> >> <FIG_VALUES>;
::= increment<HIGH _KEY> >> <FIG_VALUES>;
::= incrementN<ALPHABET> >> <FIG_VALUES>;
::= incrementN<LOW_KEY> >> <FIG_VALUES>;
::= incrementNu<ALPHABET> >> <FIG_VALUES>;
::= incrementNu<LOW_KEY> >> <FIG_VALUES>;
::= incrementNum >> <FIG_VALUES>;
::= incrementNum >> <DIGIT>+;
::= incrementNum >> <NON_ZERO>*;
::= incrementNum >> 4<NON_ZERO>*;
::= incrementNum >> 42;

Rightmost Derivation

::= <STEPWISE_STATEMENT>
::= <INCREMENT_STATEMENT>;
::= <VAR> >> <FIG_VALUES>;
::= <VAR> >> <FIG_VALUES>;
::= <VAR> >> <DIGIT>+;
::= <VAR> >> <NON_ZERO>*;
::= <VAR> >> <NON_ZERO>2;
::= <VAR> >> 42;
::= <IDEN> >> 42;
::= <ALPHABET> >> 42;
::= <LOW_KEY> >> 42;
::= <ALPHABET>m >> 42;
::= <LOW_KEY>m >> 42;
::= <ALPHABET>um >> 42;
::= <HIGH_KEY>um >> 42;
::= <ALPHABET>Num >> 42;
::= <LOW_KEY>Num >> 42;
::= <ALPHABET>tNum >> 42;
::= <LOW_KEY>tNum >> 42;
::= <ALPHABET>ntNum >> 42;
::= <LOW_KEY>ntNum >> 42;
::= <ALPHABET>entNum >> 42;
::= <LOW_KEY>entNum >> 42;
::= <ALPHABET>mentNum >> 42;
::= <LOW_KEY>mentNum >> 42;
::= <ALPHABET>ementNum >> 42;

::= <LOW_KEY>ementNum >> 42;
::= <ALPHABET>rementNum >> 42;
::= <LOW_KEY>rementNum >> 42;
::= <ALPHABET>crementNum >> 42;
::= <LOW_KEY>crementNum >> 42;
::= <ALPHABET>ncrementNum >> 42;
::= <LOW_KEY>ncrementNum >> 42;
::= incrementNum >> 42;

```
count << 5;
```

Leftmost Derivation

::= <STEPWISE_STATEMENT>
::= <DECREMENT_STATEMENT>;
::= <IDEN> << <FIG_VALUES>;
::= <ALPHABET> << <FIG_VALUES>;
::= <LOW_KEY> << <FIG_VALUES>;
::= c< ALPHABET > << <FIG_VALUES>;
::= c<LOW_KEY> << <FIG_VALUES>;
::= co<ALPHABET> << <FIG_VALUES>;
::= co<LOW_KEY> << <FIG_VALUES>;
::= cou<ALPHABET> << <FIG_VALUES>;
::= cou< LOW_KEY > << <FIG_VALUES>;
::= coun<ALPHABET> << <FIG_VALUES>;
::= coun<LOW_KEY> << <FIG_VALUES>;
::= count << <FIG_VALUES>;
::= count << <DIGIT>+;
::= count << <NON_ZERO>;
::= count << 5;

Rightmost Derivation

::= <DECREMENT_STATEMENT>;
::= <IDEN> << <FIG_VALUES>;
::= <IDEN> << <DIGIT>+;
::= <IDEN> << <NON_ZERO>;
::= <IDEN> << 5;
::= <ALPHABET> << 5;
::= <LOW_KEY> << 5;

::= <ALPHABET>t << 5;
::= <LOW_KEY>t << 5;
::= <ALPHABET>nt << 5;
::= <LOWKEY>nt << 5;
::= <ALPHABET>unt << 5;
::= <LOW_KEY >unt << 5;
::= <ALPHABET>ount << 5;
::= <LOW_KEY>ount << 5;
::= count << 5;

```
inventory >> loot;
```

Leftmost Derivation

::= <STEPWISE_STATEMENT>
::= <INCREMENT_STATEMENT>;
::= <IDEN> >> <EXPRESSION>;
::= <ALPHABET> >> <EXPRESSION>;
::= <LOW_KEY> >> <EXPRESSION>;
::= i< ALPHABET> >> <EXPRESSION>;
::= i<LOW_KEY> >> <EXPRESSION>;
::= in<ALPHABET> >> <EXPRESSION>;
::= in<LOW_KEY> >> <EXPRESSION>;
::= inv<ALPHABET> >> <EXPRESSION>;
::= inv< LOW_KEY > >> <EXPRESSION>;
::= inve<ALPHABET> >> <EXPRESSION>;
::= inve<LOW_KEY> >> <EXPRESSION>;
::= inven<ALPHABET> >> <EXPRESSION>;
::= inven<LOW_KEY> >> <EXPRESSION>;
::= invent<ALPHABET> >> <EXPRESSION>;
::= invent<LOW_KEY> >> <EXPRESSION>;
::= invento<ALPHABET> >> <EXPRESSION>;
::= invento<LOW_KEY> >> <EXPRESSION>;
::= inventor<ALPHABET> >> <EXPRESSION>;
::= inventor<LOW_KEY> >> <EXPRESSION>;
::= inventory >> <EXPRESSION>;
::= inventory >> <IDEN>;
::= inventory >> <ALPHABET>;
::= inventory >> <LOW_KEY>;
::= inventory >> l<ALPHABET>;
::= inventory >> l<LOW_KEY>;

::= inventory >> lo<ALPHABET>;
::= inventory >> lo<LOW_KEY>;
::= inventory >> loo<ALPHABET>;
::= inventory >> loo<LOW_KEY>;
::= inventory >> loot;

Rightmost Derivation

::= <STEPWISE_STATEMENT>
::= <INCREMENT_STATEMENT>;
::= <IDEN> >> <EXPRESSION>;
::= <IDEN> >> <IDEN>;
::= <IDEN> >> <ALPHABET>;
::= <IDEN> >> <LOW_KEY>;
::= <IDEN> >> <ALPHABET>t;
::= <IDEN> >> <LOW_KEY>t;
::= <IDEN> >> <ALPHABET>ot;
::= <IDEN> >> <LOW_KEY>ot;
::= <IDEN> >> <ALPHABET>oot;
::= <IDEN> >> <LOW_KEY>oot;
::= <IDEN> >> loot;
::= <ALPHABET> >> loot;
::= <LOW_KEY> >> loot;
::= < ALPHABET>y >> loot;
::= <LOW_KEY>y >> loot;
::= <ALPHABET>ry >> loot;
::= <LOW_KEY>ry >> loot;
::= <ALPHABET>ory >> loot;
::= <LOW_KEY>ory >> loot;
::= <ALPHABET>tory >> loot;
::= <LOW_KEY>tory >> loot;
::= <ALPHABET>ntory >> loot;
::= <LOW_KEY>ntory >> loot;
::= <ALPHABET>entory >> loot;
::= <LOW_KEY>entory >> loot;
::= <ALPHABET>ventory >> loot;
::= <LOW_KEY>ventory >> loot;
::= <ALPHABET>nventory >> loot;
::= <LOW_KEY>nventory >> loot;
::= inventory >> loot;

FUNCTION CONTRACTORS

<**ROUTINE_CONTRACTORS**> ::= “BET” (<CONDITION>, “<YARN>”);

```
bet(y == 5, "y equal 5");
```

Leftmost Derivatation

::= <ROUTINE_CONTRACTORS>;
::= bet (<CONDITION>, “<YARN_VALUES>”);
::= bet (<COND_EXPR>, “<YARN_VALUES>”);
::= bet (<IDEN> <COND_SYMBOL> <VALUES>, “<YARN_VALUES>”);
::= bet (<ALPHABET> <COND_SYMBOL> <VALUES>, “<YARN_VALUES>”);
::= bet (<LOW_KEY> <COND_SYMBOL> <VALUES>, “<YARN_VALUES>”);
::= bet (y <COND_SYMBOL> <VALUES>, “<YARN_VALUES>”);
::= bet (y == <VALUES>, “<YARN_VALUES>”);
::= bet (y == <FIG_VALUES>, “<YARN_VALUES>”);
::= bet (y == <NON_ZERO>, “<YARN_VALUES>”);
::= bet (y == 5, “<YARN_VALUES>”);
::= bet (y == 5, “<YARN_CONTENT>”);
::= bet (y == 5, “<ALPHABET>”);
::= bet (y == 5, “<LOW_KEY>”);
::= bet (y == 5, “y<ALPHABET>”);
::= bet (y == 5, “y<LOW_KEY>”);
::= bet (y == 5, “y e<ALPHABET>”);
::= bet (y == 5, “y e<LOW_KEY>”);
::= bet (y == 5, “y eq<ALPHABET>”);
::= bet (y == 5, “y eq<LOW_KEY>”);
::= bet (y == 5, “y equ<ALPHABET>”);
::= bet (y == 5, “y equ<LOW_KEY>”);
::= bet (y == 5, “y equa<ALPHABET>”);
::= bet (y == 5, “y equa<LOW_KEY>”);
::= bet (y == 5, “y equal<FIG_VALUES>”);
::= bet (y == 5, “y equal<NON_ZERO>”);
::= bet (y == 5, “y equal 5”);

Rightmost Derivation

::= <ROUTINE_CONTRACTORS>;
::= (<CONDITION>, “<YARN_VALUES>”);
::= (<CONDITION>, “<YARN_CONTENT>”);
::= (<CONDITION>, “<DIGIT>”);
::= (<CONDITION>, “<NON_ZERO>”);
::= (<CONDITION>, “<>5”);
::= (<CONDITION>, “<ALPHABET>5”);
::= (<CONDITION>, “<LOW_KEY>5”);
::= (<CONDITION>, “<ALPHABET>l 5”);
::= (<CONDITION>, “<LOW_KEY>l 5”);
::= (<CONDITION>, “<ALPHABET>al 5”);
::= (<CONDITION>, “<LOW_KEY>al 5”);
::= (<CONDITION>, “<ALPHABET>ual 5”);
::= (<CONDITION>, “<LOW_KEY>ual 5”);
::= (<CONDITION>, “<ALPHABET>qual 5”);
::= (<CONDITION>, “<LOW_KEY>qual 5”);
::= (<CONDITION>, “<ALPHABET>equal 5”);
::= (<CONDITION>, “<LOW_KEY>equal 5”);
::= (<CONDITION>, “y equal 5”);
::= (<COND_EXPR>, “y equal 5”);
::= (<IDEN> <COND_SYMBOL> <VALUES>, “y equal 5”);
::= (<IDEN> <COND_SYMBOL> <FIG_VALUES>, “y equal 5”);
::= (<IDEN> <COND_SYMBOL> <NON_ZERO>, “y equal 5”);
::= (<IDEN> <COND_SYMBOL> 5, “y equal 5”);
::= (<IDEN> == 5, “y equal 5”);
::= (<ALPHABET> == 5, “y equal 5”);
::= (<LOW_KEY> == 5, “y equal 5”);
::= (y == 5, “y equal 5”);
::= bet (y == 5, “y equal 5”);

```
bet (you != good, "skill issue");
```

Leftmost Derivation

::= <ROUTINE_CONTRACTORS>;
::= bet (<CONDITION>, “<YARN_VALUES>”);
::= bet (<COND_EXPR>, “<YARN_VALUES>”);
::= bet (<IDEN> <COND_SYMBOL> <VALUES>, “<YARN_VALUES>”);

::= bet (<ALPHABET> <COND_SYMBOL> <VALUES>, “<YARN_VALUES>”);
::= bet (<LOW_KEY> <COND_SYMBOL> <VALUES>, “<YARN_VALUES>”);
::= bet (y<ALPHABET> <COND_SYMBOL> <VALUES>, “<YARN_VALUES>”);
::= bet(y<LOW_KEY> <COND_SYMBOL> <VALUES>, “<YARN_VALUES>”);
::= bet (yo<ALPHABET> <COND_SYMBOL> <VALUES>, “<YARN_VALUES>”);
::= bet (yo<LOW_KEY> <COND_SYMBOL> <VALUES>, “<YARN_VALUES>”);
::= bet (you <COND_SYMBOL> <VALUES>, “<YARN_VALUES>”);
::= bet (you != <VALUES>, “<YARN_VALUES>”);
::= bet (you != <YARN_VALUES>, “<YARN_VALUES>”);
::= bet (you != <YARN_CONTENT>, “<YARN_VALUES>”);
::= bet (you != <ALPHABET>, “<YARN_VALUES>”);
::= bet (you != <LOW_KEY>, “<YARN_VALUES>”);
::= bet (you != g<ALPHABET>, “<YARN_VALUES>”);
::= bet (you != g<LOW_KEY>, “<YARN_VALUES>”);
::= bet (you != go<ALPHABET>, “<YARN_VALUES>”);
::= bet (you != go<LOW_KEY>, “<YARN_VALUES>”);
::= bet (you != goo<ALPHABET>, “<YARN_VALUES>”);
::= bet (you != goo<LOW_KEY>, “<YARN_VALUES>”);
::= bet (you != good, “<YARN_VALUES>”);
::= bet (you != good, “<YARN_CONTENT>”);
::= bet (you != good, “<ALPHABET>”);
::= bet (you != good, “<LOW_KEY>”);
::= bet (you != good, “s<ALPHABET>”);
::= bet (you != good, “s<LOW_KEY>”);
::= bet (you != good, “sk<ALPHABET>”);
::= bet (you != good, “sk<LOW_KEY>”);
::= bet (you != good, “ski<ALPHABET>”);
::= bet (you != good, “ski<LOW_KEY>”);
::= bet (you != good, “skil<ALPHABET>”);
::= bet (you != good, “skil<LOW_KEY>”);
::= bet (you != good, “skill<ALPHABET>”);
::= bet (you != good, “skill<LOW_KEY>”);
::= bet(you != good, “skill i<ALPHABET>”);
::= bet (you != good, “skill i<LOW_KEY>”);
::= bet (you != good, “skill is<ALPHABET>”);
::= bet (you != good, “skill is<LOW_KEY>”);
::= bet (you != good, “skill iss<ALPHABET>”);
::= bet(you != good, “skill iss<LOW_KEY>”);
::= bet (you != good, “skill issu<ALPHABET>”);
::= bet (you != good, “skill issu<LOW_KEY>”);
::= bet (you != good, “skill issue”);

Rightmost Derivation

::= <ROUTINE_CONTRACTORS>;
::= (<CONDITION>, “<YARN_VALUES>”);
::= (<CONDITION>, “<YARN_CONTENT>”);
::= (<CONDITION>, “<ALPHABET>”);
::= (<CONDITION>, “<LOW_KEY>”);
::= (<CONDITION>, “<ALPHABET>e”);
::= (<CONDITION>, “<LOW_KEY>e”);
::= (<CONDITION>, “<ALPHABET>ue”);
::= (<CONDITION>, “<LOW_KEY>ue”);
::= (<CONDITION>, “<ALPHABET>sue”);
::= (<CONDITION>, “<LOW_KEY>sue”);
::= (<CONDITION>, “<ALPHABET>ssue”);
::= (<CONDITION>, “<LOW_KEY>ssue”);
::= (<CONDITION>, “<ALPHABET>issue”);
::= (<CONDITION>, “<LOW_KEY>issue”);
::= (<CONDITION>, “<ALPHABET>l issue”);
::= (<CONDITION>, “<LOW_KEY>l issue”);
::= (<CONDITION>, “<ALPHABET>ll issue”);
::= (<CONDITION>, “<LOW_KEY>ll issue”);
::= (<CONDITION>, “<ALPHABET>ill issue”);
::= (<CONDITION>, “<LOW_KEY>ill issue”);
::= (<CONDITION>, “<ALPHABET>kill issue”);
::= (<CONDITION>, “<LOW_KEY>kill issue”);
::= (<CONDITION>, “skill issue”);
::= (<COND_EXPR>, “skill issue”);
::= (<IDEN> <COND_SYMBOL> <VALUES>, “skill issue”);
::= (<IDEN> <COND_SYMBOL> <YARN_VALUES>, “skill issue”);
::= (<IDEN> <COND_SYMBOL> <YARN_CONTENT>, “skill issue”);
::= (<IDEN> <COND_SYMBOL> <ALPHABET>, “skill issue”);
::= (<IDEN> <COND_SYMBOL> <LOW_KEY>, “skill issue”);
::= (<IDEN> <COND_SYMBOL> <ALPHABET>d, “skill issue”);
::= (<IDEN> <COND_SYMBOL> <LOW_KEY>d, “skill issue”);
::= (<IDEN> <COND_SYMBOL> <ALPHABET>od, “skill issue”);
::= (<IDEN> <COND_SYMBOL> <LOW_KEY>od, “skill issue”);
::= (<IDEN> <COND_SYMBOL> <ALPHABET>ood, “skill issue”);
::= (<IDEN> <COND_SYMBOL> <LOW_KEY>ood, “skill issue”);
::= (<IDEN> <COND_SYMBOL> good, “skill issue”);
::= (<IDEN> != good, “skill issue”);

::= (<ALPHABET> != good, “skill issue”);
::= (<LOW_KEY> != good, “skill issue”);
::= (<ALPHABET>u != good, “skill issue”);
::= (<LOW_KEY>u != good, “skill issue”);
::= (<ALPHABET>ou != good, “skill issue”);
::= (<LOW_KEY>ou != good, “skill issue”);
::= (you != good, “skill issue”);
::= bet (you != good, “skill issue”);

```
bet (1 > D, "huh??");
```

Leftmost Derivation

::= <ROUTINE_CONTRACTORS>;
::= bet (<CONDITION>, “<YARN_VALUES>”);
::= bet (<COND_EXPR>, “<YARN_VALUES>”);
::= bet (<IDEN> <COND_SYMBOL> <VALUES>, “<YARN_VALUES>”);
::= bet (<DIGIT> <COND_SYMBOL> <VALUES>, “<YARN_VALUES>”);
::= bet (<NON_ZERO> <COND_SYMBOL> <VALUES>, “<YARN_VALUES>”);
::= bet(1 <COND_SYMBOL> <VALUES>, “<YARN_VALUES>”);
::= bet (1 > <VALUES>, “<YARN_VALUES>”);
::= bet (1 > <YARN_VALUES>, “<YARN_VALUES>”);
::= bet (1 > <YARN_CONTENT>, “<YARN_VALUES>”);
::= bet (1 > <ALPHABET>, “<YARN_VALUES>”);
::= bet (1 > <HIGH_KEY>, “<YARN_VALUES>”);
::= bet (1 > D, “<YARN_VALUES>”);
::= bet (1 > D, “<YARN_CONTENT>”);
::= bet (1 > D, “<ALPHABET>”);
::= bet (1 > D, “<LOW_KEY>”);
::= bet (1 > D, “h<ALPHABET>”);
::= bet (1 > D, “h<LOW_KEY>”);
::= bet (1 > D, “hu<ALPHABET>”);
::= bet (1 > D, “hu<LOW_KEY>”);
::= bet(1 > D, “huh<SPECIAL_CHARS>”);
::= bet(1 > D, “huh<OTHER_SUMBOLS>”);
::= bet (1 > D, “huh?<OTHER_SUMBOLS>”);
::= bet (1 > D, “huh?<SPECIAL_CHARS>”);
::= bet (1 > D, “huh?<OTHER_SYMBOLS>”);

::= bet (1 > D, “huh??”);

Rightmost Derivation

::= <ROUTINE_CONTRACTORS>;
::= (<CONDITION>, “<YARN_VALUES>”);
::= (<CONDITION>, “<YARN_CONTENT>”);
::= (<CONDITION>, “<SPECIAL_CHARS>”);
::= (<CONDITION>, “<OTHER_SYMBOLS>”);
::= (<CONDITION>, “<SPECIAL_CHARS>?”);
::= (<CONDITION>, “<OTHER_SYMBOLS>?”);
::= (<CONDITION>, “<ALPHABET>??”);
::= (<CONDITION>, “<LOW_KEY>??”);
::= (<CONDITION>, “<ALPHABET>h??”);
::= (<CONDITION>, “<LOW_KEY>h??”);
::= (<CONDITION>, “<ALPHABET>uh??”);
::= (<CONDITION>, “<LOW_KEY>uh??”);
::= (<CONDITION>, “huh??”);
::= (<COND_EXPR>, “huh??”);
::= (<IDEN> <COND_SYMBOL> <VALUES>, “huh??”);
::= (<IDEN> <COND_SYMBOL> <YARN_VALUES>, “huh??”);
::= (<IDEN> <COND_SYMBOL> <YARN_CONTENT>, “huh??”);
::= (<IDEN> <COND_SYMBOL> <ALPHABET>, “huh??”);
::= (<IDEN> <COND_SYMBOL> <HIGH_KEY>, “huh??”);
::= (<IDEN> <COND_SYMBOL> D, “huh??”);
::= (<IDEN> > D, “huh??”);
::= (<DIGIT> > D, “huh??”);
::= (<NON_ZERO> > D, “huh??”);
::= (1 > D, “huh??”);
::= bet (1 > D, “huh??”);

DYNAMIC CALLBACK

<**CALLBACK_ROUTINE_DECLARATION**> ::= "DELAY" “ROUTINE” <ROUTINE_NAME>
"(" <PARAMETER_LIST> ")" <CODE_BLOCK>

<**CALLBACK_INVOCATION**> ::= "CHILL" <ROUTINE_NAME> "(" <ARGUMENT_LIST> ")"

<**ROUTINE_NAME**> ::= <IDEN>

<**PARAMETER_LIST**> ::= <PARAMS> ("," <PARAMETER_LIST>)*

<**ARGUMENT_LIST**> ::= <EXPRESSION> ("," <ARGUMENT_LIST>)*

<**CODE_BLOCK**> ::= "{" <STATEMENT_LIST> "}"

<**STATEMENT_LIST**> ::= <STATEMENT>+

```
delay routine buy(a1, a2) { yeet (bucks > 10) {
flex("Checkout item?");}}
```

Leftmost Derivation

::= <**CALLBACK_ROUTINE_DECLARATION**>
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" <CODE_BLOCK>
::= delay routine <IDEN> "(" <PARAMETER_LIST> ")" <CODE_BLOCK>
::= delay routine <ALPHABET> "(" <PARAMETER_LIST> ")" <CODE_BLOCK>
::= delay routine <LOW_KEY> "(" <PARAMETER_LIST> ")" <CODE_BLOCK>
::= delay routine b<ALPHABET> "(" <PARAMETER_LIST> ")" <CODE_BLOCK>
::= delay routine b<LOW_KEY> "(" <PARAMETER_LIST> ")" <CODE_BLOCK>
::= delay routine bu<ALPHABET> "(" <PARAMETER_LIST> ")" <CODE_BLOCK>
::= delay routine bu<LOW_KEY> "(" <PARAMETER_LIST> ")" <CODE_BLOCK>
::= delay routine buy "(" <PARAMETER_LIST> ")" <CODE_BLOCK>
::= delay routine buy (<PARAMETER_LIST> ")" <CODE_BLOCK>
::= delay routine buy (<PARAMS> ("," <PARAMETER_LIST>)* ")" <CODE_BLOCK>
::= delay routine buy (<IDEN>* (, IDEN)* ("," <PARAMETER_LIST>)* ")" <CODE_BLOCK>
::= delay routine buy (<ALPHABET> ("," <PARAMETER_LIST>)* ")" <CODE_BLOCK>
::= delay routine buy (<LOW_KEY> ("," <PARAMETER_LIST>)* ")" <CODE_BLOCK>
::= delay routine buy (a<DIGIT> ("," <PARAMETER_LIST>)* ")" <CODE_BLOCK>
::= delay routine buy (a<NON_ZERO> ("," <PARAMETER_LIST>)* ")" <CODE_BLOCK>
::= delay routine buy (a1 , <PARAMETER_LIST>)* ")" <CODE_BLOCK>

::= delay routine buy (a1 , <PARAMS> ")" <CODE_BLOCK>
::= delay routine buy (a1 , <IDEN> ")" <CODE_BLOCK>
::= delay routine buy (a1 , <ALPHABET> ")" <CODE_BLOCK>
::= delay routine buy (a1 , <LOW_KEY> ")" <CODE_BLOCK>
::= delay routine buy (a1 , a<DIGIT> ")" <CODE_BLOCK>
::= delay routine buy (a1 , a<NON_ZERO> ")" <CODE_BLOCK>
::= delay routine buy (a1 , a2 ")" <CODE_BLOCK>
::= delay routine buy (a1 , a2)"{" <STATEMENT_LIST> "}"
::= delay routine buy (a1 , a2){ <STATEMENT>+ "}"
::= delay routine buy (a1 , a2){ <PROG_STMT>+ "}"
::= delay routine buy (a1 , a2){ <COND_STMT> "}"
::= delay routine buy (a1 , a2){ <YEET> "}"
::= delay routine buy (a1 , a2){ “yeet “(” <COND_LOGIC_EXPR> “){“ <PROG_STMT> “}”
"}"
::= delay routine buy (a1 , a2){ yeet (<IDEN> <COND_SYMBOL> <VALUES> “){“
<PROG_STMT> “}” "}"
::= delay routine buy (a1 , a2){ yeet (<ALPHABET> <COND_SYMBOL> <VALUES> “){“
<PROG_STMT> “}” "}"
::= delay routine buy (a1 , a2){ yeet (<LOW_KEY> <COND_SYMBOL> <VALUES> “){“
<PROG_STMT> “}” "}"
::= delay routine buy (a1 , a2){ yeet (b<LOW_KEY> <COND_SYMBOL> <VALUES> “){“
<PROG_STMT> “}” "}"
::= delay routine buy (a1 , a2){ yeet (bu<LOW_KEY> <COND_SYMBOL> <VALUES> “){“
<PROG_STMT> “}” "}"
::= delay routine buy (a1 , a2){ yeet (buc<LOW_KEY> <COND_SYMBOL> <VALUES> “){“
<PROG_STMT> “}” "}"
::= delay routine buy (a1 , a2){ yeet (buck<LOW_KEY> <COND_SYMBOL> <VALUES>
“){“ <PROG_STMT> “}” "}"
::= delay routine buy (a1 , a2){ yeet (bucks <COND_SYMBOL> <VALUES> “){“
<PROG_STMT> “}” "}"
::= delay routine buy (a1 , a2){ yeet (bucks > <VALUES> “){“ <PROG_STMT> “}” "}"
::= delay routine buy (a1 , a2){ yeet (bucks > <FIG_VALUES> “){“ <PROG_STMT> “}” "}"
::= delay routine buy (a1 , a2){ yeet (bucks > <DIGIT>+ “){“ <PROG_STMT> “}” "}"
::= delay routine buy (a1 , a2){ yeet (bucks > <NON_ZERO> “){“ <PROG_STMT> “}” "}"
::= delay routine buy (a1 , a2){ yeet (bucks > 1<NON_ZERO> “){“ <PROG_STMT> “}” "}"
::= delay routine buy (a1 , a2){ yeet (bucks > 10){ <PROG_STMT> “}” "}"
::= delay routine buy (a1 , a2){ yeet (bucks > 10){ <OUT_STMTS> “}” "}"
::= delay routine buy (a1 , a2){ yeet (bucks > 10){ flex “(“<YARN_VALUES>“)”;“}” "}"
::= delay routine buy (a1 , a2){ yeet (bucks > 10){ flex ("Checkout item?");“}”"}"
::= delay routine buy (a1 , a2){ yeet (bucks > 10){ flex ("Checkout item?"); } }

Rightmost Derivation

::= <**CALLBACK_ROUTINE_DECLARATION**>
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" <CODE_BLOCK>
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" {" <STATEMENT_LIST> }
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" {" <STATEMENT>+ }
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" {" <PROG_STMT>+ }
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" { <COND_STMT> }
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" { <YEET> }
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" { “yeet “(”
<COND_LOGIC_EXPR> “){“ <PROG_STMT> } }
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" { “yeet “(”
<COND_LOGIC_EXPR> “){“ <OUT_STMTS> } }
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" { “yeet “(”
<COND_LOGIC_EXPR> “){“ flex “(“ <YARN_VALUES> “)”; } }
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" { “yeet “(”
<COND_LOGIC_EXPR> “){“ flex ("Checkout item?"); } }
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" { “yeet “(” <IDEN>
<COND_SYMBOL> <VALUES> “){“ flex ("Checkout item?"); } }
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" { “yeet “(” <IDEN>
<COND_SYMBOL> <FIG_VALUES>){ flex ("Checkout item?"); } }
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" { “yeet “(” <IDEN>
<COND_SYMBOL> <DIGIT>+){ flex ("Checkout item?"); } }
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" { “yeet “(” <IDEN>
<COND_SYMBOL> <NON_ZERO>){ flex ("Checkout item?"); } }
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" { “yeet “(” <IDEN>
<COND_SYMBOL> <NON_ZERO>0){ flex ("Checkout item?"); } }
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" { “yeet “(” <IDEN>
<COND_SYMBOL> 10){ flex ("Checkout item?"); } }
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" { “yeet “(” <IDEN> > 10
){ flex ("Checkout item?"); } }
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" { “yeet “(” <ALPHABET>
> 10){ flex ("Checkout item?"); } }
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" { “yeet “(” <LOW_KEY>
> 10){ flex ("Checkout item?"); } }
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" { “yeet “(” <LOW_KEY>s
> 10){ flex ("Checkout item?"); } }
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" { “yeet “(”
<LOW_KEY>ks > 10){ flex ("Checkout item?"); } }

::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" { “yeet “(” <LOW_KEY>cks > 10){ flex ("Checkout item?"); } }
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" { “yeet “(” <LOW_KEY>ucks > 10){ flex ("Checkout item?"); } }
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST>) { yeet (bucks > 10){ flex ("Checkout item?"); } }
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST>) { yeet (bucks > 10){ flex ("Checkout item?"); } }
::= delay routine <ROUTINE_NAME> "(" <PARAMS> ("," <PARAMETER_LIST>)*) { yeet (bucks > 10){ flex ("Checkout item?"); } }
::= delay routine <ROUTINE_NAME> "(" <PARAMS> ","<PARAMS>) { yeet (bucks > 10){ flex ("Checkout item?"); } }
::= delay routine <ROUTINE_NAME> "(" <PARAMS> ","<IDEN>) { yeet (bucks > 10){ flex ("Checkout item?"); } }
::= delay routine <ROUTINE_NAME> "(" <PARAMS> ","<DIGIT>) { yeet (bucks > 10){ flex ("Checkout item?"); } }
::= delay routine <ROUTINE_NAME> "(" <PARAMS> ","<NON_ZERO>) { yeet (bucks > 10){ flex ("Checkout item?"); } }
::= delay routine <ROUTINE_NAME> "(" <PARAMS> ","<ALPHABET>2) { yeet (bucks > 10){ flex ("Checkout item?"); } }
::= delay routine <ROUTINE_NAME> "(" <PARAMS> ","<LOW_KEY>2) { yeet (bucks > 10){ flex ("Checkout item?"); } }
::= delay routine <ROUTINE_NAME> "(" <PARAMS> ,a2) { yeet (bucks > 10){ flex ("Checkout item?"); } }
::= delay routine <ROUTINE_NAME> "(" <IDEN> ,a2) { yeet (bucks > 10){ flex ("Checkout item?"); } }
::= delay routine <ROUTINE_NAME> "(" <DIGIT> ,a2) { yeet (bucks > 10){ flex ("Checkout item?"); } }
::= delay routine <ROUTINE_NAME> "(" <NON_ZERO> ,a2) { yeet (bucks > 10){ flex ("Checkout item?"); } }
::= delay routine <ROUTINE_NAME> "(" <ALPHABET>1,a2) { yeet (bucks > 10){ flex ("Checkout item?"); } }
::= delay routine <ROUTINE_NAME> "(" <LOW_KEY>1,a2) { yeet (bucks > 10){ flex ("Checkout item?"); } }
::= delay routine <ROUTINE_NAME> (a1,a2) { yeet (bucks > 10){ flex ("Checkout item?"); } }
::= delay routine <IDEN> (a1,a2) { yeet (bucks > 10){ flex ("Checkout item?"); } }
::= delay routine <ALPHABET> (a1,a2) { yeet (bucks > 10){ flex ("Checkout item?"); } }
::= delay routine <LOW_KEY> (a1,a2) { yeet (bucks > 10){ flex ("Checkout item?"); } }
::= delay routine <ALPHABET>y (a1,a2) { yeet (bucks > 10){ flex ("Checkout item?"); } }
::= delay routine <LOW_KEY>y (a1,a2) { yeet (bucks > 10){ flex ("Checkout item?"); } }

::= delay routine <ALPHABET>uy (a1,a2) { yeet (bucks > 10){ flex ("Checkout item?"); }
}
::= delay routine <LOW_KEY>uy (a1,a2) { yeet (bucks > 10){ flex ("Checkout item?"); } }
::= delay routine buy (a1,a2) { yeet (bucks > 10){ flex ("Checkout item?"); } }

```
chill funcA ();
```

Leftmost Derivation

::= **<CALLBACK_INVOCATION>**
::= chill <ROUTINE_NAME> "(" <ARGUMENT_LIST> ")";
::= chill <IDEN> "(" <ARGUMENT_LIST> ")";
::= chill <ALPHABET> "(" <ARGUMENT_LIST> ")";
::= chill <LOW_KEY> "(" <ARGUMENT_LIST> ")";
::= chill f<ALPHABET> "(" <ARGUMENT_LIST> ")";
::= chill f<LOW_KEY> "(" <ARGUMENT_LIST> ")";
::= chill fu<ALPHABET> "(" <ARGUMENT_LIST> ")";
::= chill fu<LOW_KEY> "(" <ARGUMENT_LIST> ")";
::= chill fun<ALPHABET> "(" <ARGUMENT_LIST> ")";
::= chill func<HIGH_KEY> "(" <ARGUMENT_LIST> ")";
::= chill funcA "(" <ARGUMENT_LIST> ")";
::= chill funcA ();

Rightmost Derivation

::= **<CALLBACK_INVOCATION>**
::= chill <ROUTINE_NAME> "(" <ARGUMENT_LIST> ")";
::= chill <ROUTINE_NAME> ();
::= chill <IDEN> ();
::= chill <ALPHABET> ();
::= chill <HIGH_KEY> ();
::= chill <ALPHABET>A ();
::= chill <LOW_KEY>A ();
::= chill <ALPHABET>cA ();
::= chill <LOW_KEY>cA ();
::= chill <ALPHABET>ncA ();
::= chill <LOW_KEY>ncA ();
::= chill <ALPHABET>uncA ();
::= chill <LOW_KEY>uncA ();

::= chill funcA ();

```
delay routine do(game, sleep){
//
}
yeet (task == 0) { chill do(game,sleep); }
```

Leftmost Derivation

::= <**CALLBACK_ROUTINE_DECLARATION**>
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" <CODE_BLOCK>
<PROG_STMT>
::= delay routine <IDEN> "(" <PARAMETER_LIST> ")" <CODE_BLOCK> <PROG_STMT>
::= delay routine <ALPHABET> "(" <PARAMETER_LIST> ")" <CODE_BLOCK>
<PROG_STMT>
::= delay routine <LOW_KEY> "(" <PARAMETER_LIST> ")" <CODE_BLOCK>
<PROG_STMT>
::= delay routine d<ALPHABET> "(" <PARAMETER_LIST> ")" <CODE_BLOCK>
<PROG_STMT>
::= delay routine d<LOW_KEY> "(" <PARAMETER_LIST> ")" <CODE_BLOCK>
<PROG_STMT>
::= delay routine do "(" <PARAMETER_LIST> ")" <CODE_BLOCK> <PROG_STMT>
::= delay routine do (<PARAMS> "," <PARAMETER_LIST>)* ")" <CODE_BLOCK>
<PROG_STMT>
::= delay routine do (<IDEN>* (, IDEN)* "," <PARAMETER_LIST>)* ")" <CODE_BLOCK>
<PROG_STMT>
::= delay routine do (<ALPHABET> "," <PARAMETER_LIST>)* ")" <CODE_BLOCK>
<PROG_STMT>
::= delay routine do (<LOW_KEY> "," <PARAMETER_LIST>)* ")" <CODE_BLOCK>
<PROG_STMT>
::= delay routine do (g<LOW_KEY> "," <PARAMETER_LIST>)* ")" <CODE_BLOCK>
<PROG_STMT>
::= delay routine do (ga<LOW_KEY> "," <PARAMETER_LIST>)* ")" <CODE_BLOCK>
<PROG_STMT>
::= delay routine do (gam<LOW_KEY> "," <PARAMETER_LIST>)* ")" <CODE_BLOCK>
<PROG_STMT>
::= delay routine do (game , <PARAMETER_LIST> ")" <CODE_BLOCK> <PROG_STMT>
::= delay routine do (game , <PARAMS> ")" <CODE_BLOCK> <PROG_STMT>

::= delay routine do (game , <IDEN> ")" <CODE_BLOCK> <PROG_STMT>
::= delay routine do (game , <ALPHABET> ")" <CODE_BLOCK> <PROG_STMT>
::= delay routine do (game , <LOW_KEY> ")" <CODE_BLOCK> <PROG_STMT>
::= delay routine do (game , s<LOW_KEY> ")" <CODE_BLOCK> <PROG_STMT>
::= delay routine do (game , sle<LOW_KEY> ")" <CODE_BLOCK> <PROG_STMT>
::= delay routine do (game , slee<LOW_KEY> ")" <CODE_BLOCK> <PROG_STMT>
::= delay routine do (game , sleep) <CODE_BLOCK> <PROG_STMT>
::= delay routine do (game , sleep) { <STATEMENT_LIST> "}" <PROG_STMT>
::= delay routine do (game , sleep) { <STATEMENT>+ "}" <PROG_STMT>
******::= delay routine do (game , sleep) { //"}" <PROG_STMT>
::= delay routine do (game , sleep) { // } <PROG_STMT>
::= delay routine do (game , sleep) { // } <COND_ STMTS>
::= delay routine do (game , sleep) { // } <YEET>
::= delay routine do (game , sleep) { // } “yeet (” <COND_LOGIC_EXPR> “){“
<PROG_STMT> “}”
::= delay routine do (game , sleep) { // } yeet (<IDEN> <COND_SYMBOL> <VALUES>
“){“ <PROG_STMT> “}”
::= delay routine do (game , sleep) { // } yeet (<ALPHABET> <COND_SYMBOL>
<VALUES> “){“ <PROG_STMT> “}”
::= delay routine do (game , sleep) { // } yeet (<LOWKEY> <COND_SYMBOL>
<VALUES> “){“ <PROG_STMT> “}”
::= delay routine do (game , sleep) { // } yeet (t<LOWKEY> <COND_SYMBOL>
<VALUES> “){“ <PROG_STMT> “}”
::= delay routine do (game , sleep) { // } yeet (ta<LOWKEY> <COND_SYMBOL>
<VALUES> “){“ <PROG_STMT> “}”
::= delay routine do (game , sleep) { // } yeet (tas<LOWKEY> <COND_SYMBOL>
<VALUES> “){“ <PROG_STMT> “}”
::= delay routine do (game , sleep) { // } yeet (task <COND_SYMBOL> <VALUES> “){“
<PROG_STMT> “}”
::= delay routine do (game , sleep) { // } yeet (task == <VALUES> “){“ <PROG_STMT>
“}”
::= delay routine do (game , sleep) { // } yeet (task == <VALUES> “){“ <PROG_STMT>
“}”
::= delay routine do (game , sleep) { // } yeet (task == <FIG_VALUES> “){“
<PROG_STMT> “}”
::= delay routine do (game , sleep) { // } yeet (task == <DIGIT> “){“ <PROG_STMT> “}”
"}"
::= delay routine do (game , sleep) { // } yeet (task == <NON_ZERO> “){“
<PROG_STMT> “}”
::= delay routine do (game , sleep) { // } yeet (task == 10){“ <PROG_STMT> “}”

::= delay routine do (game , sleep) { // } yeet (task == 10){“
<CALLBACK_INVOCATION> “}”
::= delay routine do (game , sleep) { // } yeet (task == 10){“ "CHILL"
<ROUTINE_NAME> "(" <ARGUMENT_LIST> ")" “;” “}”
::= delay routine do (game , sleep) { // } yeet (task == 10){ chill <IDEN> "("
<ARGUMENT_LIST> ")" “;” “}”
::= delay routine do (game , sleep) { // } yeet (task == 10){ chill <ALPHABET> "("
<ARGUMENT_LIST> ")" “;” “}”
::= delay routine do (game , sleep) { // } yeet (task == 10){ chill <LOW_KEY> "("
<ARGUMENT_LIST> ")" “;” “}”
::= delay routine do (game , sleep) { // } yeet (task == 10){ chill d<LOW_KEY> (
<ARGUMENT_LIST> ")" “;” “}”
::= delay routine do (game , sleep) { // } yeet (task == 10){ chill do (<EXPRESSION> ")"
“;” “}”
::= delay routine do (game , sleep) { // } yeet (task == 10){ chill do (
<PARAMETER_LIST> ")" “;” “}”
::= delay routine do (game , sleep) { // } yeet (task == 10){ chill do (<PARAMS> ")" “;”
“}”

::= delay routine do (game , sleep) { // } yeet (task == 10){ chill do (<IDEN> ")" “;” “}”
::= delay routine do (game , sleep) { // } yeet (task == 10){ chill do (<IDEN>*")" “;” “}”
::= delay routine do (game , sleep) { // } yeet (task == 10){ chill do (<ALPHABET> ”,”
<IDEN> ")" “;” “}”
::= delay routine do (game , sleep) { // } yeet (task == 10){ chill do (<LOW_KEY> ”,”
<IDEN> ")" “;” “}”
::= delay routine do (game , sleep) { // } yeet (task == 10){ chill do (g<LOW_KEY> ”,”
<IDEN> ")" “;” “}”
::= delay routine do (game , sleep) { // } yeet (task == 10){ chill do (ga<LOW_KEY> ”,”
<IDEN> ")" “;” “}”
::= delay routine do (game , sleep) { // } yeet (task == 10){ chill do (gam<LOW_KEY>
”,” <IDEN> ")" “;” “}”
::= delay routine do (game , sleep) { // } yeet (task == 10){ chill do (game , <IDEN> ")"
“;” “}”
::= delay routine do (game , sleep) { // } yeet (task == 10){ chill do (game ,
<ALPHABET> ")" “;” “}”
::= delay routine do (game , sleep) { // } yeet (task == 10){ chill do (game ,
<LOW_KEY> ")" “;” “}”
::= delay routine do (game , sleep) { // } yeet (task == 10){ chill do (game ,
s<LOW_KEY> ")" “;” “}”
::= delay routine do (game , sleep) { // } yeet (task == 10){ chill do (game ,
sl<LOW_KEY> ")" “;” “}”

::= delay routine do (game , sleep) { // } yeet (task == 10){ chill do (game , sle<LOW_KEY> ")" ";" "}"
::= delay routine do (game , sleep) { // } yeet (task == 10){ chill do (game , slee<LOW_KEY> ")" ";" "}"
::= delay routine do (game , sleep) { // } yeet (task == 10){ chill do (game , sleep); }

Rightmost Derivation

::= <**CALLBACK_ROUTINE_DECLARATION**>
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" <CODE_BLOCK>
<PROG_STMT> <PROG_STMT>
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" <CODE_BLOCK>
<PROG_STMT> <CALLBACK_INVOCATION>
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" <CODE_BLOCK>
<PROG_STMT> {" chill <ROUTINE_NAME> "(" <ARGUMENT_LIST> ")" ";" "}"
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" <CODE_BLOCK>
<PROG_STMT> {" chill <ROUTINE_NAME> "(" <EXPRESSION> ")" ";" "}"
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" <CODE_BLOCK>
<PROG_STMT> {" chill <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" ";" "}"
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" <CODE_BLOCK>
<PROG_STMT> {" chill <ROUTINE_NAME> "(" <PARAMS> ")" ";" "}"
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" <CODE_BLOCK>
<PROG_STMT> {" chill <ROUTINE_NAME> "(" <IDEN> ")" ";" "}"
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" <CODE_BLOCK>
<PROG_STMT> {" chill <ROUTINE_NAME> "(" <IDEN> "," <ALPHABET> ")" ";" "}"
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" <CODE_BLOCK>
<PROG_STMT> {" chill <ROUTINE_NAME> "(" <IDEN> "," <LOW_KEY> ")" ";" "}"
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" <CODE_BLOCK>
<PROG_STMT> {" chill <ROUTINE_NAME> "(" <IDEN> "," <ALPHABET>p ")" ";" "}"
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" <CODE_BLOCK>
<PROG_STMT> {" chill <ROUTINE_NAME> "(" <IDEN> "," <LOW_KEY>p ")" ";" "}"
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" <CODE_BLOCK>
<PROG_STMT> {" chill <ROUTINE_NAME> "(" <IDEN> "," <LOW_KEY>ep ")" ";" "}"
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" <CODE_BLOCK>
<PROG_STMT> {" chill <ROUTINE_NAME> "(" <IDEN> "," <LOW_KEY>eep ")" ";" "}"
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" <CODE_BLOCK>
<PROG_STMT> {" chill <ROUTINE_NAME> "(" <IDEN> "," <LOW_KEY>leep ")" ";" "}"

::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" <CODE_BLOCK>
<PROG_STMT> {“ chill <ROUTINE_NAME> "(" <IDEN> ”,” sleep ")" “;” “}”
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" <CODE_BLOCK>
<PROG_STMT> {“ chill <ROUTINE_NAME> "(" <ALPHABET> ”,” sleep ")" “;” “}”
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" <CODE_BLOCK>
<PROG_STMT> {“ chill <ROUTINE_NAME> "(" <LOW_KEY> ”,” sleep ")" “;” “}”
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" <CODE_BLOCK>
<PROG_STMT> {“ chill <ROUTINE_NAME> "(" <LOW_KEY>e ”,” sleep ")" “;” “}”
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" <CODE_BLOCK>
<PROG_STMT> {“ chill <ROUTINE_NAME> "(" <LOW_KEY>me ”,” sleep ")" “;” “}”
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" <CODE_BLOCK>
<PROG_STMT> {“ chill <ROUTINE_NAME> "(" <LOW_KEY>ame ”,” sleep ")" “;” “}”
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" <CODE_BLOCK>
<PROG_STMT> {“ chill <ROUTINE_NAME> (game , sleep) ; }

::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" <CODE_BLOCK>
<PROG_STMT> {“ chill <IDEN> (game , sleep) ; }
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" <CODE_BLOCK>
<PROG_STMT> {“ chill <ALPHABET> (game , sleep) ; }
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" <CODE_BLOCK>
<PROG_STMT> {“ chill <LOW_KEY> (game , sleep) ; }
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" <CODE_BLOCK>
<PROG_STMT> {“ chill <LOW_KEY>o (game , sleep) ; }
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" <CODE_BLOCK>
<PROG_STMT> {“ chill do (game , sleep) ; }
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" <COND_STMTS> {“ chill
do (game , sleep) ; }
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" <CODE_BLOCK>
<YEET> { chill do (game , sleep) ; }
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" <CODE_BLOCK>
<YEET> { chill do (game , sleep) ; }
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" <CODE_BLOCK> yeet {
chill do (game , sleep) ; }
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" <STATEMENT_LIST>
yeet { chill do (game , sleep) ; }
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" <STATEMENT>+ yeet {
chill do (game , sleep) ; }
::= delay routine <ROUTINE_NAME> "(" <PARAMETER_LIST> ")" {//} yeet { chill do (
game , sleep) ; }
::= delay routine <ROUTINE_NAME> "(" <PARAMS> ")" {//} yeet { chill do (game , sleep
) ; }

::= delay routine <ROUTINE_NAME> "(" <IDEN> ")" {//} yeet { chill do (game , sleep) ; }
::= delay routine <ROUTINE_NAME> "(" <IDEN> ,<ALPHABET> ")" {//} yeet { chill do (game , sleep) ; }
::= delay routine <ROUTINE_NAME> "(" <IDEN>, <LOW_KEY> ")" {//} yeet { chill do (game , sleep) ; }
::= delay routine <ROUTINE_NAME> "(" <IDEN>, <LOW_KEY>p ")" {//} yeet { chill do (game , sleep) ; }
::= delay routine <ROUTINE_NAME> "(" <IDEN>, <LOW_KEY>ep ")" {//} yeet { chill do (game , sleep) ; }
::= delay routine <ROUTINE_NAME> "(" <IDEN>, <LOW_KEY>eep ")" {//} yeet { chill do (game , sleep) ; }
::= delay routine <ROUTINE_NAME> "(" <IDEN>, <LOW_KEY>leep ")" {//} yeet { chill do (game , sleep) ; }
::= delay routine <ROUTINE_NAME> "(" <IDEN>, sleep ")" {//} yeet { chill do (game , sleep) ; }
::= delay routine <ROUTINE_NAME> "(" <ALPHABET>, sleep ")" {//} yeet { chill do (game , sleep) ; }
::= delay routine <ROUTINE_NAME> "(" <LOW_KEY>, sleep ")" {//} yeet { chill do (game , sleep) ; }
::= delay routine <ROUTINE_NAME> "(" <LOW_KEY>e, sleep ")" {//} yeet { chill do (game , sleep) ; }
::= delay routine <ROUTINE_NAME> "(" <LOW_KEY>me, sleep ")" {//} yeet { chill do (game , sleep) ; }
::= delay routine <ROUTINE_NAME> "(" <LOW_KEY>ame, sleep ")" {//} yeet { chill do (game , sleep) ; }
::= delay routine <ROUTINE_NAME> (game, sleep) {//} yeet { chill do (game , sleep) ; }
::= delay routine <IDEN> (game, sleep) {//} yeet { chill do (game , sleep) ; }
::= delay routine <ALPHABET> (game, sleep) {//} yeet { chill do (game , sleep) ; }
::= delay routine <LOW_KEY> (game, sleep) {//} yeet { chill do (game , sleep) ; }
::= delay routine <LOW_KEY>o (game, sleep) {//} yeet { chill do (game , sleep) ; }
::= delay routine do (game, sleep) {//} yeet { chill do (game , sleep) ; }

LOOPING STATEMENTS

<**RELAPSE**> ::= “relapse (worse:” <FIG_VALUE> “;recover:” <FIG_VALUE> “) {“ <STMT> “)” | “relapse (as:” <IDEN> “;worse:” <FIG_VALUE> “;recover:” <FIG_VALUE> “) {“ <STMT> “)”

```
relapse (worse: 3; recover: 9) {}
```

Leftmost Derivation

::= <RELAPSE>
::= relapse “(worse:” <FIG_VALUE> “; recover:” <FIG_VALUE> “)” “{” “}”
::= relapse (worse: <FIG_VALUES> “; recover:” <FIG_VALUE> “)” “{” “}”
::= relapse (worse: <DIGIT> “; recover:” <FIG_VALUE> “)” “{” “}”
::= relapse (worse: <NON_ZERO> “; recover:” <FIG_VALUE> “)” “{” “}”
::= relapse (worse: 3; “recover”: <FIG_VALUE> “)” “{” “}”
::= relapse (worse: 3; recover: <FIG_VALUE> “)” “{” “}”
::= relapse (worse: 3; recover: <DIGIT> “)” “{” “}”
::= relapse (worse: 3; recover: <NONE_ZERO> “)” “{” “}”
::= relapse (worse: 3; recover: 9 “)” “{” “}”
::= relapse (worse: 3; recover: 9) “{” “}”
::= relapse (worse: 3; recover: 9) { “}”
::= relapse (worse: 3; recover: 9) {}

Rightmost Derivation

::= <RELAPSE>
::= relapse “(worse:” <FIG_VALUE> “; recover:” <FIG_VALUE> “)” “{” “}”
::= relapse “(worse:” <FIG_VALUE> “; recover:” <FIG_VALUE> “)” “{”}
::= relapse “(worse:” <FIG_VALUE> “; recover:” <FIG_VALUE> “)” {}
::= relapse “(worse:” <FIG_VALUE> “; recover:” <FIG_VALUE>) {}
::= relapse “(worse:” <FIG_VALUE> “; recover:” <DIGIT>) {}
::= relapse “(worse:” <FIG_VALUE> “; recover:” <NONE_ZERO>) {}
::= relapse “(worse:” <FIG_VALUE> “; recover:” 9) {}
::= relapse “(worse:” <FIG_VALUE>; recover: 9) {}
::= relapse “(worse:” <DIGIT>; recover: 9) {}
::= relapse “(worse:” <NON_ZERO>; recover: 9) {}
::= relapse “(worse:” 3; recover: 9) {}
::= relapse (worse: 3; recover: 9) {}

```
relapse (as: y; worse: 1; recover: 9) {}
```

Leftmost Derivation

::= <RELAPSE>
::= relapse “(as:” <IDEN> “; worse:” <FIG_VALUE> “; recover:” <FIG_VALUE> “)” “{” “}”
::= relapse (as: <IDEN> “; worse:” <FIG_VALUE> “; recover:” <FIG_VALUE> “)” “{” “}”
::= relapse (as: <ALPHABET> “; worse:” <FIG_VALUE> “; recover:” <FIG_VALUE> “)” “{”
“}”
::= relapse (as: <LOW_KEY> “; worse:” <FIG_VALUE> “; recover:” <FIG_VALUE> “)” “{” “}”
::= relapse (as: y “; worse:” <FIG_VALUE> “; recover:” <FIG_VALUE> “)” “{” “}”
::= relapse (as: y; worse: <FIG_VALUE> “; recover:” <FIG_VALUE> “)” “{” “}”
::= relapse (as: y; worse: <DIGIT> “; recover:” <FIG_VALUE> “)” “{” “}”
::= relapse (as: y; worse: <NO_ZERO> “; recover:” <FIG_VALUE> “)” “{” “}”
::= relapse (as: y; worse: 1 “; recover:” <FIG_VALUE> “)” “{” “}”
::= relapse (as: y; worse: 1; recover: <FIG_VALUE> “)” “{” “}”
::= relapse (as: y; worse: 1; recover: <DIGIT> “)” “{” “}”
::= relapse (as: y; worse: 1; recover: <NON_ZERO> “)” “{” “}”
::= relapse (as: y; worse: 1; recover: 9 “)” “{” “}”
::= relapse (as: y; worse: 1; recover: 9) “{” “}”
::= relapse (as: y; worse: 1; recover: 9) { “}”
::= relapse (as: y; worse: 1; recover: 9) {}

Rightmost Derivation

::= <RELAPSE>
::= relapse “(as:” <IDEN> “; worse:” <FIG_VALUE> “; recover:” <FIG_VALUE> “)” “{” “}”
::= relapse “(as:” <IDEN> “; worse:” <FIG_VALUE> “; recover:” <FIG_VALUE> “)” “{” }
::= relapse “(as:” <IDEN> “; worse:” <FIG_VALUE> “; recover:” <FIG_VALUE> “)” {}
::= relapse “(as:” <IDEN> “; worse:” <FIG_VALUE> “; recover:” <FIG_VALUE>) {}
::= relapse “(as:” <IDEN> “; worse:” <FIG_VALUE> “; recover:” <DIGIT>) {}
::= relapse “(as:” <IDEN> “; worse:” <FIG_VALUE> “; recover:” <NON_ZERO>) {}
::= relapse “(as:” <IDEN> “; worse:” <FIG_VALUE> “; recover:” 9) {}
::= relapse “(as:” <IDEN> “; worse:” <FIG_VALUE>; recover: 9) {}
::= relapse “(as:” <IDEN> “; worse:” <DIGIT>; recover: 9) {}
::= relapse “(as:” <IDEN> “; worse:” <NON_ZERO>; recover: 9) {}
::= relapse “(as:” <IDEN> “; worse:” 1; recover: 9) {}
::= relapse “(as:” <IDEN>; worse: 1; recover: 9) {}
::= relapse “(as:” <ALPHABET>; worse: 1; recover: 9) {}

::= relapse “(as:” <LOW_KEY>; worse: 1; recover: 9) {}
::= relapse “(as:” y; worse: 1; recover: 9) {}
::= relapse (as: y; worse: 1; recover: 9) {}

relapse (worse: 1; recover: 20) {}

Leftmost Derivation

::= <RELAPSE>
::= relapse “(worse:” <FIG_VALUE> “; recover:” <FIG_VALUE> “)” “{” “}”
::= relapse (worse: <FIG_VALUES> “; recover:” <FIG_VALUE> “)” “{” “}”
::= relapse (worse: <DIGIT> “; recover:” <FIG_VALUE> “)” “{” “}”
::= relapse (worse: <NON_ZERO> “; recover:” <FIG_VALUE> “)” “{” “}”
::= relapse (worse: 1; “recover”: <FIG_VALUE> “)” “{” “}”
::= relapse (worse: 1; recover: <FIG_VALUE> “)” “{” “}”
::= relapse (worse: 1; recover: <DIGIT> “)” “{” “}”
::= relapse (worse: 1; recover: <NONE_ZERO> “)” “{” “}”
::= relapse (worse: 1; recover: 2<FIG_VALUE> “)” “{” “}”
::= relapse (worse: 1; recover: 2<DIGIT> “)” “{” “}”
::= relapse (worse: 1; recover: 20 “)” “{” “}”
::= relapse (worse: 1; recover:) “{” “}”
::= relapse (worse: 1; recover: 20) { “}”
::= relapse (worse: 1; recover: 20) {}

Rightmost Derivation

::= <RELAPSE>
::= relapse “(worse:” <FIG_VALUE> “; recover:” <FIG_VALUE> “)” “{” “}”
::= relapse “(worse:” <FIG_VALUE> “; recover:” <FIG_VALUE> “)” “{”}
::= relapse “(worse:” <FIG_VALUE> “; recover:” <FIG_VALUE> “)” {}
::= relapse “(worse:” <FIG_VALUE> “; recover:” <FIG_VALUE>) {}
::= relapse “(worse:” <FIG_VALUE> “; recover:” <DIGIT>) {}
::= relapse “(worse:” <FIG_VALUE> “; recover:” <NONE_ZERO>) {}
::= relapse “(worse:” <FIG_VALUE> “; recover:” 2<FIG_VALUE>) {}
::= relapse “(worse:” <FIG_VALUE> “; recover:” 2<DIGIT>) {}
::= relapse “(worse:” <FIG_VALUE> “; recover:” 20) {}
::= relapse “(worse:” <FIG_VALUE>; recover: 20) {}
::= relapse “(worse:” <DIGIT>; recover: 20) {}
::= relapse “(worse:” <NON_ZERO>; recover: 20) {}

::= relapse “(worse:” 1; recover: 20) {}
::= relapse (worse: 1; recover: 20) {}

WEBDEV STATEMENTS

<HTML_SUPPORT> ::= “htmlize” { <HTML_COMPONENT> }

<**HTML_COMPONENT**> ::= <HTML_TAGS> { “<YARN>” : “<YARN>”} |
(<HTML_COMPONENT>,)*

<**HTML_TAGS**> ::= "<body>", "<h1>", "<h2>", "<h3>", <h4>, <h5>, <h6>, "<p>", "<a>",
"<img>", " ", "", "<ul>", "<ol>", "<li>", "
", "<hr>", "", "",
"<blockquote>", "<cite>", "<code>", "<pre>", "<i>", "<b>", "<u>", "<small>", "<sub>",
"<sup>", "<abbr>", "<address>", "<var>", "<samp>", "<header>", "<nav>", "<main>",
"<section>", "<article>", "<aside>", "<footer>", "<address>", "<a>", "",
"", "<small>", "<s>", "<cite>", "<q>", "<dfn>", "<abbr>", "<data>", "<time>",
"<code>", "<var>", "<samp>", "<kbd>", "<sub>", "<sup>", "<i>", "<b>", "<u>",
"<mark>", "<ruby>", "", "
", "<img>", "<iframe>", "<embed>", "<param>",
"<video>", "<audio>", "<source>", "<track>", "<map>", "<area>", "<a>", "<table>",
"<caption>", "<colgroup>", "<col>", "<thead>", "<tbody>", "<tfoot>", "<tr>", "<th>",
"<td>", "<form>", "<label>", "<input>", "<button>", "<select>", "<datalist>",
"<optgroup>", "<option>", "<textarea>", "<output>", "<progress>", "<meter>",
"<fieldset>", "<legend>", "<details>", "<summary>", "<dialog>", "<script>",
"<noscript>", "<template>", "<slot>", "<canvas>", "<svg>", "<math>",”<content>”

```
htmlize{ h1 { content: “Beans”}}
```

Leftmost Derivation

::= <HTML_SUPPORT> {<HTML_COMPONENT>}
::= htmlize {HTML_COMPONENT}

::= htmlize {<HTML_TAGS> { <HTML_TAGS> : “<YARN_VALUES>”}}
::= htmlize {h1 { <HTML_TAGS> : “<YARN_VALUES>”}}
::= htmlize {h1 { content : “<YARN_VALUES>”}}
::= htmlize {h1 { content : “<YARN_CONTENT>”}}
::= htmlize {h1 { content : “<ALPHABET>”}}
::= htmlize {h1 { content : “<HIGH_KEY>”}}
::= htmlize {h1 { content : “B<ALPHABET>”}}
::= htmlize {h1 { content : “B<LOW_KEY>”}}
::= htmlize {h1 { content : “Be<ALPHABET>”}}
::= htmlize {h1 { content : “Be<LOW_KEY>”}}
::= htmlize {h1 { content : “Bea<ALPHABET>”}}
::= htmlize {h1 { content : “Bea<LOW_KEY>”}}
::= htmlize {h1 { content : “Bean<ALPHABET>”}}
::= htmlize {h1 { content : “Bean<LOW_KEY>”}}
::= htmlize {h1 { content : “Beans”}}

Rightmost Derivation

::= <HTML_SUPPORT> {<HTML_COMPONENT>}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<YARN_VALUES>”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<YARN_CONTENT>”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<ALPHABET>”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<LOW_KEY>”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<ALPHABET>s”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<LOW_KEY>s”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<ALPHABET>ns”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<LOWKEY>ns”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<ALPHABET>ans”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<LOW_KEY>ans”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<ALPHABET>eans”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<HIGH_KEY>eans”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “Beans”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { content : “Beans”}}
::= <HTML_SUPPORT> {<h1> { content : “Beans”}}
::= htmlize {<h1> { content : “Beans”}}

```
htmlize{ h6 {content: “BSCS 3-1N”} body
{content: “Vince”}
```

Leftmost Derivation

::= <HTML_SUPPORT> {<HTML_COMPONENT>}
::= htmlize {HTML_COMPONENT}
::= htmlize {<HTML_TAGS> { <HTML_TAGS> : “<YARN_VALUES>”} <HTML_TAGS> {
<HTML_TAGS> : “<YARN_VALUES>”}}
::=htmlize {h6 {<HTML_TAGS> : “<YARN_VALUES>”} <HTML_TAGS> { <HTML_TAGS> :
“<YARN_VALUES>” }}
::=htmlize {h6 { content : “<YARN_VALUES>”} <HTML_TAGS> { <HTML_TAGS> :
“<YARN_VALUES>” }}
::=htmlize {h6 { content : “<YARN_CONTENT>”} <HTML_TAGS> { <HTML_TAGS> :
“<YARN_VALUES>” }}

::=htmlize {h6 { content : “<ALPHABET>”} <HTML_TAGS> { <HTML_TAGS> :
“<YARN_VALUES>” }}
::=htmlize {h6 { content : “<HIGH_KEY>”} <HTML_TAGS> { <HTML_TAGS> :
“<YARN_VALUES>” }}
::=htmlize {h6 { content : “B<YARN_CONTENT>”} <HTML_TAGS> { <HTML_TAGS> :
“<YARN_VALUES>” }}
::=htmlize {h6 { content : “B<HIGH_KEY>”} <HTML_TAGS> { <HTML_TAGS> :
“<YARN_VALUES>” }}
::=htmlize {h6 { content : “BS<ALPHABET>”} <HTML_TAGS> { <HTML_TAGS> :
“<YARN_VALUES>” }}
::=htmlize {h6 { content : “BS<HIGH_KEY>”} <HTML_TAGS> { <HTML_TAGS> :
“<YARN_VALUES>” }}
::=htmlize {h6 { content : “BSC<ALPHABET>”} <HTML_TAGS> { <HTML_TAGS> :
“<YARN_VALUES>” }}
::=htmlize {h6 { content : “BSC<HIGH_KEY>”} <HTML_TAGS> { <HTML_TAGS> :
“<YARN_VALUES>” }}
::=htmlize {h6 { content : “BSCS <FIG_VALUES>”} <HTML_TAGS> { <HTML_TAGS> :
“<YARN_VALUES>” }}
::=htmlize {h6 { content : “BSCS <DIGIT>”} <HTML_TAGS> { <HTML_TAGS> :
“<YARN_VALUES>” }}
::=htmlize {h6 { content : “BSCS <NON_ZERO>”} <HTML_TAGS> { <HTML_TAGS> :
“<YARN_VALUES>” }}
::=htmlize {h6 { content : “BSCS 3<YARN_CONTENT>”} <HTML_TAGS> { <HTML_TAGS> :
“<YARN_VALUES>” }}
::=htmlize {h6 { content : “BSCS 3<SPECIAL_CHARS>”} <HTML_TAGS> { <HTML_TAGS> :
“<YARN_VALUES>” }}
::=htmlize {h6 { content : “BSCS 3-<FIG_VALUES>”} <HTML_TAGS> { <HTML_TAGS> :
“<YARN_VALUES>” }}
::=htmlize {h6 { content : “BSCS 3-<DIGIT>”} <HTML_TAGS> { <HTML_TAGS> :
“<YARN_VALUES>” }}
::=htmlize {h6 { content : “BSCS 3-<NON_ZERO>”} <HTML_TAGS> { <HTML_TAGS> :
“<YARN_VALUES>” }}
::=htmlize {h6 { content : “BSCS 3-1<YARN_CONTENT>”} <HTML_TAGS> { <HTML_TAGS>
: “<YARN_VALUES>” }}

::=htmlize {h6 { content : “BSCS 3-1<ALPHABET>”} <HTML_TAGS> { <HTML_TAGS> :
“<YARN_VALUES>” }}
::=htmlize {h6 { content : “BSCS 3-1<HIGH_KEY>”} <HTML_TAGS> { <HTML_TAGS> :
“<YARN_VALUES>” }}
::=htmlize {h6 { content : “BSCS 3-1N”} <HTML_TAGS> { <HTML_TAGS> :
“<YARN_VALUES>” }}
::=htmlize {h6 { content : “BSCS 3-1N”} body { <HTML_TAGS> : “<YARN_VALUES>” }}
::=htmlize {h6 { content : “BSCS 3-1N”} body { <HTML_TAGS> : “<YARN_VALUES>” }}
::=htmlize {h6 { content : “BSCS 3-1N”} body {content: “<YARN_VALUES>” }}
::=htmlize {h6 { content : “BSCS 3-1N”} body {content: “<YARN_CONTENT>” }}
::=htmlize {h6 { content : “BSCS 3-1N”} body {content: “<ALPHABET>” }}
::=htmlize {h6 { content : “BSCS 3-1N”} body {content: “<HIGH_KEY>” }}
::=htmlize {h6 { content : “BSCS 3-1N”} body {content: “V<ALPHABET>” }}
::=htmlize {h6 { content : “BSCS 3-1N”} body {content: “V<LOW_KEY>” }}
::=htmlize {h6 { content : “BSCS 3-1N”} body {content: “Vi<ALPHABET>” }}
::=htmlize {h6 { content : “BSCS 3-1N”} body {content: “Vi<LOW_KEY>” }}
::=htmlize {h6 { content : “BSCS 3-1N”} body {content: “Vin<ALPHABET>” }}
::=htmlize {h6 { content : “BSCS 3-1N”} body {content: “Vin<LOW_KEY>” }}
::=htmlize {h6 { content : “BSCS 3-1N”} body {content: “Vinc<ALPHABET>” }}
::=htmlize {h6 { content : “BSCS 3-1N”} body {content: “Vinc<LOW_KEY>” }}
::=htmlize {h6 { content : “BSCS 3-1N”} body {content: “Vince” }}

Rightmost Derivation

::= <HTML_SUPPORT> {<HTML_COMPONENT>}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<YARN_VALUES>”}
<HTML_TAGS> { <HTML_TAGS> : “<YARN_VALUES>”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<YARN_VALUES>”}
<HTML_TAGS> { <HTML_TAGS> : “<YARN_CONTENT>”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<YARN_VALUES>”}
<HTML_TAGS> { <HTML_TAGS> : “<ALPHABET>”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<YARN_VALUES>”}
<HTML_TAGS> { <HTML_TAGS> : “<LOW_KEY>”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<YARN_VALUES>”}
<HTML_TAGS> { <HTML_TAGS> : “<ALPHABET>e”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<YARN_VALUES>”}
<HTML_TAGS> { <HTML_TAGS> : “<LOW_KEY>e”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<YARN_VALUES>”}
<HTML_TAGS> { <HTML_TAGS> : “<ALPHABET>ce”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<YARN_VALUES>”}
<HTML_TAGS> { <HTML_TAGS> : “<LOW_KEY>ce”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<YARN_VALUES>”}
<HTML_TAGS> { <HTML_TAGS> : “<ALPHABET>nce”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<YARN_VALUES>”}
<HTML_TAGS> { <HTML_TAGS> : “<LOW_KEY>nce”}}

::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<YARN_VALUES>”}
<HTML_TAGS> { <HTML_TAGS> : “<ALPHABET>ince”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<YARN_VALUES>”}
<HTML_TAGS> { <HTML_TAGS> : “<HIGH_KEY>ince”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<YARN_VALUES>”}
<HTML_TAGS> { <HTML_TAGS> : “Vince”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<YARN_VALUES>”}
<HTML_TAGS> { content : “Vince”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<YARN_VALUES>”} body {
content : “Vince”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<YARN_VALUES>”} body {
content : “Vince”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<CHAR_VALUE>”} body {
content : “Vince”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<ALPHABET>”} body { content
: “Vince”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<HIGH_KEY>”} body { content
: “Vince”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<FIG_VALUES>N”} body {
content : “Vince”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<DIGIT>N”} body { content :
“Vince”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<NON_ZERO>N”} body {
content : “Vince”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<YARN_CONTENT>1N”} body {
content : “Vince”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<SPECIAL_CHARS>1N”} body
{ content : “Vince”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<FIG_VALUES>-1N”} body {
content : “Vince”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<DIGIT>-1N”} body { content :
“Vince”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<NON_ZERO>-1N”} body {
content : “Vince”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<CHAR_VALUE>3-1N”} body {
content : “Vince”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<ALPHABET>3-1N”} body {
content : “Vince”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<HIGH_KEY>3-1N”} body {
content : “Vince”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<ALPHABET>S 3-1N”} body {
content : “Vince”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<HIGH_KEY>S 3-1N”} body {
content : “Vince”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<ALPHABET>CS 3-1N”} body {
content : “Vince”}}

::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<HIGH_KEY>CS 3-1N”} body {
content : “Vince”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<ALPHABET>SCS 3-1N”} body
{ content : “Vince”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<HIGH_KEY>CS 3-1N”} body {
content : “Vince”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “BSCS 3-1N”} body { content :
“Vince”}}
::= <HTML_SUPPORT> {<HTML_TAGS> { content : “BSCS 3-1N”} body { content :
“Vince”}}
::= <HTML_SUPPORT> { h6 { content : “BSCS 3-1N”} body { content : “Vince”}}
::= htmlize { h6 { content : “BSCS 3-1N”} body { content : “Vince”}}

```
htmlize {h1{id: “Six” content: “One”}}
```

Leftmost Derivation

::= <HTML_SUPPORT> {<HTML_COMPONENT>}
::= htmlize{<HTML_COMPONENT>}
::= htmlize {<HTML_TAGS> { <HTML_TAGS> : “<YARN_VALUES>” <HTML_TAGS> :
“<YARN_VALUES>” }}
::= htmlize { h1 { <HTML_TAGS> : “<YARN_VALUES>” <HTML_TAGS> : “<YARN_VALUES>”
}}
::= htmlize { h1 { id : “<YARN_VALUES>” <HTML_TAGS> : “<YARN_VALUES>” }}
::= htmlize { h1 { id : “<YARN_VALUES>” <HTML_TAGS> : “<YARN_VALUES>” }}
::= htmlize { h1 { id : “<YARN_CONTENT>” <HTML_TAGS> : “<YARN_VALUES>” }}
::= htmlize { h1 { id : “<ALPHABET>” <HTML_TAGS> : “<YARN_VALUES>” }}
::= htmlize { h1 { id : “<HIGH_KEY>” <HTML_TAGS> : “<YARN_VALUES>” }}
::= htmlize { h1 { id : “S<ALPHABET>” <HTML_TAGS> : “<YARN_VALUES>” }}
::= htmlize { h1 { id : “S<LOW_KEY>” <HTML_TAGS> : “<YARN_VALUES>” }}
::= htmlize { h1 { id : “Si<ALPHABET>” <HTML_TAGS> : “<YARN_VALUES>” }}
::= htmlize { h1 { id : “Si<LOW_KEY>” <HTML_TAGS> : “<YARN_VALUES>” }}
::= htmlize { h1 { id : “Six” <HTML_TAGS> : “<YARN_VALUES>” }}
::= htmlize { h1 { id : “Six” content : “<YARN_VALUES>” }}
::= htmlize { h1 { id : “Six” content : “<YARN_CONTENT>” }}
::= htmlize { h1 { id : “Six” content : “<ALPHABET>” }}
::= htmlize { h1 { id : “Six” content : “<HIGH_KEY>” }}
::= htmlize { h1 { id : “Six” content : “O<ALPHABET>” }}
::= htmlize { h1 { id : “Six” content : “O<LOW_KEY>” }}
::= htmlize { h1 { id : “Six” content : “On<ALPHABET>” }}
::= htmlize { h1 { id : “Six” content : “On<LOW_KEY>” }}

::= htmlize { h1 { id : “Six” content : “One” }}

Rightmost Derivation

::= <HTML_SUPPORT> {<HTML_COMPONENT>}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<YARN_VALUES>”
<HTML_TAGS> : “<YARN_VALUES>” }}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<YARN_VALUES>”
<HTML_TAGS> : “<YARN_CONTENT>” }}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<YARN_VALUES>”
<HTML_TAGS> : “<ALPHABET>” }}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<YARN_VALUES>”
<HTML_TAGS> : “<LOW_KEY>” }}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<YARN_VALUES>”
<HTML_TAGS> : “<ALPHABET>e” }}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<YARN_VALUES>”
<HTML_TAGS> : “<LOW_KEY>e” }}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<YARN_VALUES>”
<HTML_TAGS> : “<ALPHABET>ne” }}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<YARN_VALUES>”
<HTML_TAGS> : “<HIGH_KEY>ne” }}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<YARN_VALUES>”
<HTML_TAGS> : “One” }}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<YARN_VALUES>” content :
“One” }}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<YARN_CONTENT>” content :
“One” }}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<ALPHABET>” content : “One”
}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<LOW_KEY>” content : “One”
}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<ALPHABET>x” content :
“One” }}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<LOW_KEY>x” content : “One”
}}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<ALPHABET>ix” content :
“One” }}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “<HIGH_KEY>ix” content :
“One” }}
::= <HTML_SUPPORT> {<HTML_TAGS> { <HTML_TAGS> : “Six” content : “One” }}
::= <HTML_SUPPORT> {<HTML_TAGS> { id : “Six” content : “One” }}

::= <HTML_SUPPORT> {h1 { id : “Six” content : “One” }}
::= htmlize {h1 { id : “Six” content : “One” }}

If You Know, You Know

Programming Language Documentation

Authors:

Alamag, Jose Luis
Calendario, Mark Kenneth
Caspe, Mark Vincent
Favorito, Vince Lennard
Lalis, Reygine
Villegas, Daniel

Web App Preview

https://iykyk-31n.vercel.app/

Repository

https://github.com/markcalendario/IYKYK-programming-language