# BoboLang

# Contents

# Chapter 1

# Terminal symbols regex

$$\texttt{ID} \quad ::= \quad \Big(\texttt{a} + \ldots + \texttt{Z} + (\texttt{\_}\,(\texttt{\_})^*(\texttt{a} + \ldots + \texttt{Z} + \texttt{0} + \ldots + \texttt{9}))\Big)\Big(\texttt{a} + \ldots + \texttt{Z} + \texttt{0} + \ldots + \texttt{9} + \texttt{\_}\Big)^*$$

# Chapter 2

# File

$$\begin{array}{rcl}
\langle \textit{file} \rangle & ::= & \langle \textit{importlist} \rangle \ \langle \textit{filep} \rangle \\[1em]
\langle \textit{importlist} \rangle & ::= & \texttt{import ID} \ \langle \textit{importidp} \rangle \ \langle \textit{importlist} \rangle \\
 & | & \varepsilon \\[1em]
\langle \textit{importidp} \rangle & ::= & \texttt{.ID} \ \langle \textit{importidp} \rangle \\
 & | & \varepsilon \\[1em]
\langle \textit{filep} \rangle & ::= & \texttt{module ID \{} \ \langle \textit{modulep} \rangle \ \texttt{\}} \\
 & | & \langle \textit{scriptp} \rangle \\[1em]
\langle \textit{modulep} \rangle & ::= & \langle \textit{assignment} \rangle \ \langle \textit{modulep} \rangle \\
 & | & \langle \textit{fun} \rangle \ \langle \textit{modulep} \rangle \\
 & | & \varepsilon \\[1em]
\langle \textit{scriptp} \rangle & ::= & \langle \textit{fun} \rangle \ \langle \textit{scriptp} \rangle \\
 & | & \langle \textit{stat} \rangle \ \langle \textit{scriptp} \rangle \\
 & | & \varepsilon
\end{array}$$

# Chapter 3

# Functions

$$\begin{array}{rcl}
\langle \mathit{fun} \rangle & ::= & \langle \mathit{funmod} \rangle \; \texttt{fun ID (} \; \langle \mathit{arglist} \rangle \; \texttt{) \{} \; \langle \mathit{statlist} \rangle \; \texttt{\}} \\
\\
\langle \mathit{funmod} \rangle & ::= & \texttt{private} \\
 & | & \texttt{public} \\
 & | & \varepsilon \\
\\
\langle \mathit{statlist} \rangle & ::= & \ldots \\
\\
\langle \mathit{arglist} \rangle & ::= & \langle \mathit{var} \rangle \; \langle \mathit{arglistp} \rangle \\
 & | & \varepsilon \\
\\
\langle \mathit{arglistp} \rangle & ::= & \texttt{,} \; \langle \mathit{var} \rangle \; \langle \mathit{arglistp} \rangle \\
 & | & \texttt{=} \; \langle \mathit{expr} \rangle \; \langle defarglistp \rangle \\
 & | & \varepsilon \\
\\
\langle defarglistp \rangle & ::= & \texttt{,} \; \langle \mathit{var} \rangle \; \texttt{=} \; \langle \mathit{expr} \rangle \; \langle defarglistp \rangle \\
 & | & \varepsilon
\end{array}$$

```
[public] fun start(const a, [var] b) {
    ...
}

fun ciao(x, y = 2) {
    ...
}
```

# Chapter 4

# Statements

## 4.1 Assignments

$$\begin{array}{rcl}
\langle \textit{var} \rangle & ::= & \langle \textit{varmod} \rangle\ \texttt{ID} \\[1em]
\langle \textit{varmod} \rangle & ::= & \texttt{const} \\
 & | & \texttt{var} \\
 & | & \varepsilon \\[1em]
\langle \textit{assignment} \rangle & ::= & \langle \textit{var} \rangle\ \langle \textit{assignmentp} \rangle\ \langle \textit{expr} \rangle \\[1em]
\langle \textit{assignmentp} \rangle & ::= & \texttt{=}\ \langle \textit{var} \rangle\ \langle \textit{assignmentp} \rangle \\
 & | & \texttt{=}
\end{array}$$

## 4.2 Consequence

$$\begin{array}{rcl}
\langle \textit{conseq} \rangle & ::= & \langle \textit{bexpr} \rangle\ \texttt{->}\ \langle \textit{statlist} \rangle
\end{array}$$

### 4.2.1

```
x <= 0 -> io.err("x must be positive")
```

### 4.2.2

```
(x <= 0) -> {
    io.err("x must be positive")
    return
}
```

## 4.3 Check

$$\begin{array}{rcl}
\langle \textit{check} \rangle & ::= & \texttt{check \{}\ \langle \textit{checklist} \rangle\ \texttt{\}} \\[1em]
\langle \textit{checklist} \rangle & ::= & \langle \textit{conseq} \rangle\ \langle \textit{checklistp} \rangle \\[1em]
\langle \textit{checklistp} \rangle & ::= & \langle \textit{conseq} \rangle\ \langle \textit{checklistp} \rangle \\
 & | & \varepsilon
\end{array}$$

### 4.3.1 Simple

```
check {
    x > 0 -> io.print("nyum")
}
```

### 4.3.2 Check, else

```
check {
    is_number(x) -> io.print("nyum")
    true         -> return
}
```

### 4.3.3 Check, else-if, else

```
check {
    is_number(x) -> io.print("nyum")
    is_string(x) -> io.print("nope")
    true         -> return
}
```

### 4.3.4 Nested usage

```
check {
    is_integer(x) -> check {
        x == 0 -> io.print("Zero")
        x < 0  -> io.print("Negative")
        x > 0  -> io.print("Positive")
    }

    is_string(x) -> ...
}
```