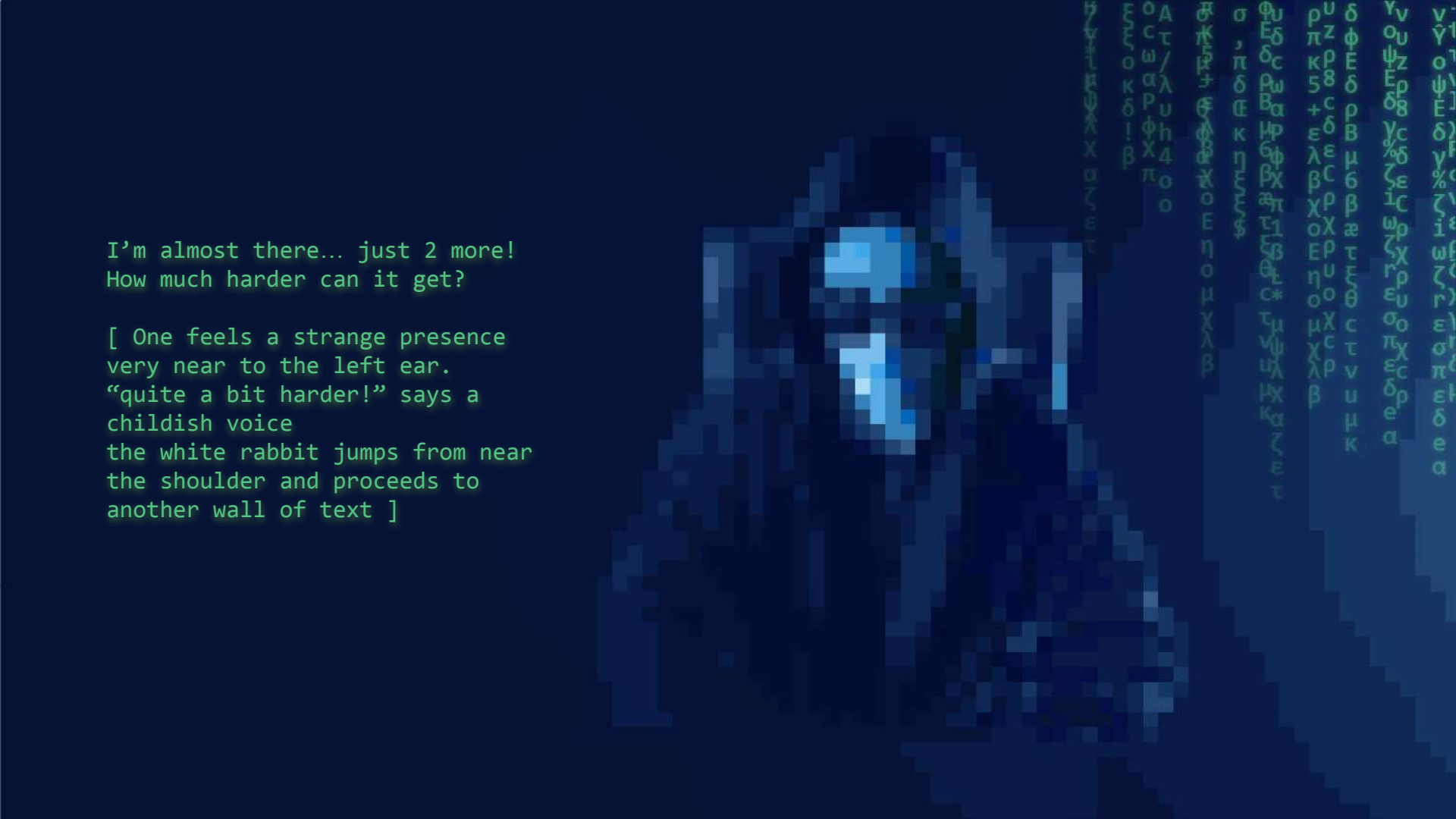

LEVEL 7


I’m almost there… just 2 more!
How much harder can it get?
[One feels a strange presence
very near to the left ear.
“quite a bit harder!” says a
childish voice
the white rabbit jumps from near
the shoulder and proceeds to
another wall of text]

New in Level 7:
Complex expressions

In this level, values used in statements can now be obtained by either arithmetic expressions or boolean expressions, or combination of both if the context allows it.

An arithmetic Expression can be one of the following:

- an **integer**
- a **variable** that contains an integer
- a **function call** that returns an integer
- **(** arithmeticExpression **)**
- arithmeticExpression **+** arithmeticExpression
- arithmeticExpression ***** arithmeticExpression

The value of any arithmetic expression will never exceed **10 to the power of 18**

If the expression **operands** for **+** or ***** cannot be resolved to integers, an error will be produced.

A boolean Expression can be one of the following:

- **true**
- **false**
- a **variable** contains an boolean
- a **function call** that returns a boolean
- **(** booleanExpression **)**
- booleanExpression **and** booleanExpression
- booleanExpression **or** booleanExpression
- arithmeticExpression **<** arithmeticExpression
- value **==** other value

For **==** the values can be **arithmeticExpression**, **booleanExpression** or **string**.

It returns **true** if both sides have the same **type** and are **equal**, otherwise it returns **false**.

If the expression operands for **and** or **or** cannot be resolved to **booleans** or the operands for **<** cannot be resolved to **integers**, an **error** will be produced and dealt with, as was described in the previous level.

Expressions are always evaluated from **right** to **left**, with respect to the precedence order, **parentheses**, **function calls**, *****, **+**, **<**, **==**, **and**, **or.**

Which also implies that the **expression** of the **right operand** always gets **resolved** but the type **check** will be done once both operands are resolved.

Considering these changes execute all the functions like in the previous levels.

The input format is identical to last level

	Input	Output
Example	34 start print 1 + 2 print call (1 + 2) print call call call 5 + 2 print call (call call 5 + 0) print 2 * (1 + 2) print true or call (call 2 * 4) try print true and call 2 + 0 end catch print errorcatched end end start return 1 end start return 3 end start return false end start return 5 end start print 1 < 2 print 2 < 1 print 1 + 2 < 1 + 3 print 1 + 2 < 1 + 3 == true print 1 + 2 < 1 + 3 == string print 1 + 2 < 1 + 3 == 4 end	33756trueerrorcatched truefalsetruetruefalsefalse

The input format is identical to last level

	Input	Output
Example	23 start try print call 2 + call 3 + call 4 * 1 + call 5 end catch end end start print two return 2 end start print three return true end start print four return 4 end start print five return 5 end	fivefourthree two three four five Hint: Notice that the first function does not execute the call of function two. That is because function three returns a boolean which causes an error when the + is evaluated.


GOOD
LUCK!