LEVEL 6

I see 3 more walls of text which
I haven’t gone through yet ...

[the white rabbit appears in
front, looks at me, nods his head
up and down then jumps towards
another panel of text]

New in Level 6:

Catching Errors

To deal with possible errors we introduce the **try** / **catch** statement.

This statement is defined as the following:

- **try** <statement>* **end catch** <statement>* **end**

The statement(s) in the catch block will only be executed if some error happened in its corresponding try block.

Similar to other languages if some error happened outside a try block, the error will propagate to the calling function and if the call statement in that function is not in a try block it will propagate to it calling function and so on.

Similar to **if** / **else**, **try** / **catch** have their own **execution queue**.

Also starting from this level, When a function calls another function, the called function has access to all the current values of the variables of it's calling function, or any of the calling functions of its calling functions and so on.

If some variables value got modified in the called function, all modifications will be **undone** after the execution of this function is done.

However a function **cannot create** variable with the same name as already created variable in any of its calling functions, doing so will produce an **error** which will propagate up if it is not in a try block.

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

Compiler Breakout

The input format is the same like last level

	Input	Output

Example 2	`13`
	`start`
	`var variable value`
	`call 2`
	`end`
	`start`
	`try`
	`print something`
	`var variable newvalue`
	`end`
	`catch`
	`print variablealreadyinitialised`
	`end`
	`end`
		`somethingvariablealreadyinitialised`
		`something`

Good
LUCK!