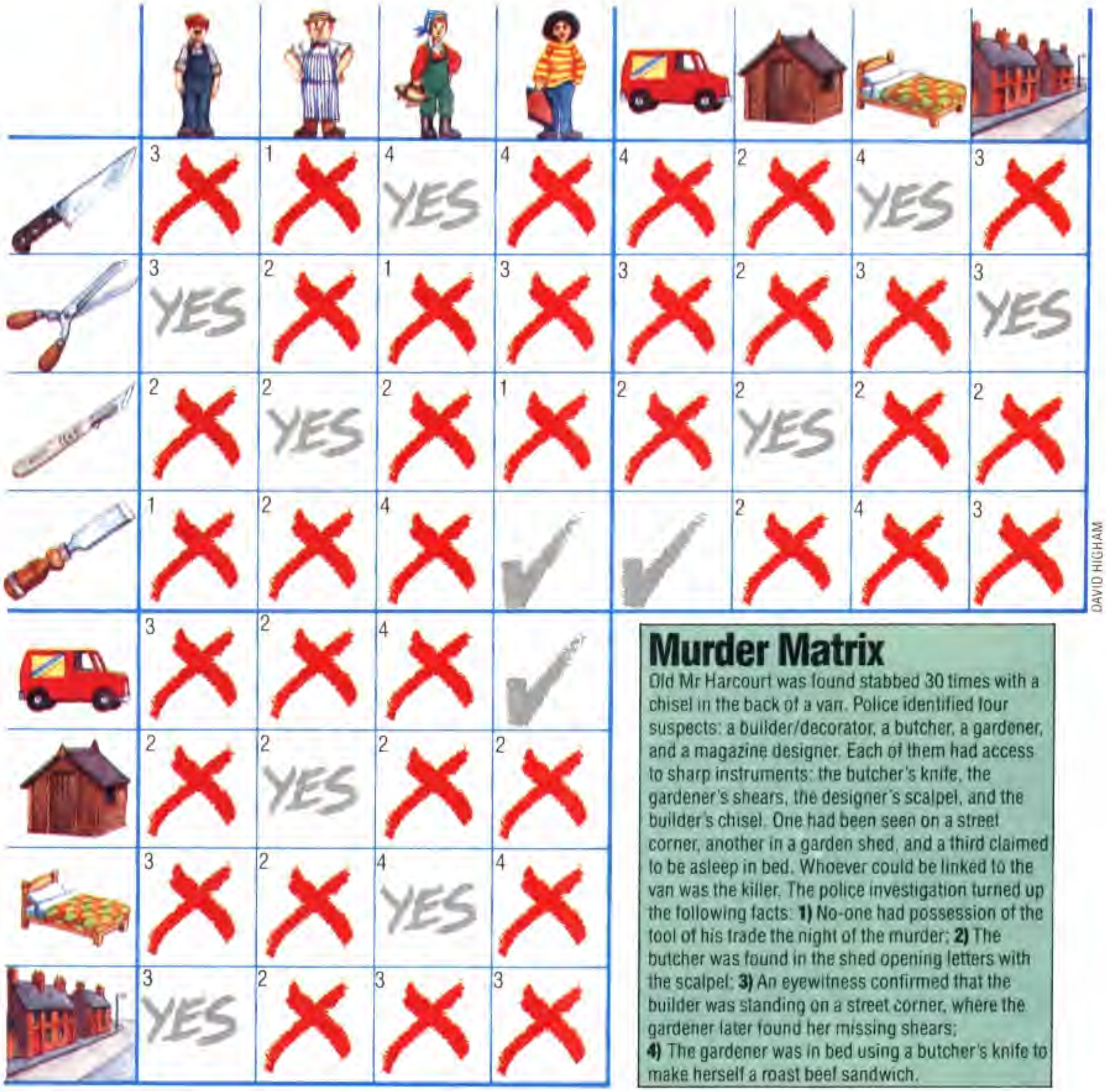

3 ✗	1 ✗	4 YES	4 ✗	4 ✗	2 ✗	4 YES	3 ✗
3 YES	2 ✗	1 ✗	3 ✗	3 ✗	2 ✗	3 ✗	3 YES
2 ✗	2 YES	2 ✗	1 ✗	2 ✗	2 YES	2 ✗	2 ✗
1 ✗	2 ✗	4 ✗	✓	✓	2 ✗	4 ✗	3 ✗
3 ✗	2 ✗	4 ✗	✓				
2 ✗	2 YES	2 ✗	2 ✗				
3 ✗	2 ✗	4 YES	4 ✗				
3 YES	2 ✗	3 ✗	3 ✗				

DAVID HIGHAM

Murder Matrix

Old Mr Harcourt was found stabbed 30 times with a chisel in the back of a van. Police identified four suspects: a builder/decorator, a butcher, a gardener, and a magazine designer. Each of them had access to sharp instruments: the butcher's knife, the gardener's shears, the designer's scalpel, and the builder's chisel. One had been seen on a street corner, another in a garden shed, and a third claimed to be asleep in bed. Whoever could be linked to the van was the killer. The police investigation turned up the following facts: **1)** No-one had possession of the tool of his trade the night of the murder; **2)** The butcher was found in the shed opening letters with the scalpel; **3)** An eyewitness confirmed that the builder was standing on a street corner, where the gardener later found her missing shears; **4)** The gardener was in bed using a butcher's knife to make herself a roast beef sandwich.

In order not to lose track of where it is up to in its assignment of variables, COMPARE puts the present values on a *stack* before MATCH? is used (since MATCH? may alter these assignments), and then recovers these values afterwards. Here is the full procedure:

```
TO COMPARE :QUERIES :DATA
   IF EMPTY? :QUERIES THEN MAKE "ANS FPUT
      :VARS :ANS STOP
   IF EMPTY? :DATA THEN STOP
   PUSH :VARS
   TEST MATCH? FIRST :QUERIES FIRST :DATA
   IFTRUE COMPARE BUTFIRST :QUERIES
      :DATABASE
   PULL "VARS
   COMPARE :QUERIES BUTFIRST :DATA
END
```

In COMPARE we use a stack to keep track of the value of VARS, instead of using a temporary variable, because COMPARE could call itself between the time we want to save the values and the time we want to restore them. Therefore, any such temporary variable could be overwritten by the next call and the original values lost. The stack prevents this from happening.

PUSH puts a value on 'top' of the stack, first creating the variable STACK if it does not already exist.

```
TO PUSH :DATA
   IF NOT THING? :STACK THEN MAKE "STACK []
   MAKE "STACK FPUT :DATA :STACK
END
```

PULL takes an item from the stack, and assigns it as the value of a variable.

```
TO PULL :NAME
   MAKE :NAME FIRST :STACK
   MAKE "STACK BUTFIRST :STACK
END
```

What we have then are the rudiments of a 'logic programming' language. That is a language in which we simply add facts and rules to a database and then query that database by means of logical descriptions of the data we require. The best example to date of a logic programming language is PROLOG — but that's another story!

Logo Flavours

Some versions of MIT LOGO do not have EMPTY? or MEMBER?. Definitions for these have been given previously (see pages 754 and 776).

In all LCSI versions use EMPTYP for EMPTY? and MEMBERP for MEMBER?. There is a primitive, EQUALP, which tests whether its two inputs are the same. Use it for comparing lists and words instead of an equals sign (which works for lists on some LCSI versions, but not on others).

The IF syntax in LCSI LOGO is demonstrated by:

```
IF EMPTYP :CONTENTS
 [PRINT [NOTHING
 SPECIAL]] [PRINT
 :CONTENTS]
```

The first list after the condition is performed if the condition is true, and the second if it is false.

LCSI LOGO also supports the TEST, IFTRUE, IFFALSE syntax for conditionals