```
ROOM as THING :ROOM.NAME.
  TO ASSIGN.VARIABLES
    MAKE "ROOM.NAME WORD "ROOM. :HERE
    MAKE "ROOM THING :ROOM.NAME
    MAKE "DESCRIPTION DESCRIPTION :ROOM
    MAKE "CONTENTS CONTENTS :ROOM
    MAKE "EXIT.LIST EXIT.LIST :ROOM
  END
```

It is now possible for you to draw up a map of the locations in your adventure game, and list the descriptions of them (with contents and exits). In the next instalment, we will conclude our general discussion by looking at movement between locations and how 'perils' are implemented. We will then begin considering a complete adventure game as an example of what can be done.

Logo Flavours

Some versions of MIT LOGO do not have EMPTY?, ITEM, COUNT or MEMBER?. Definitions for the first three were given in the last instalment. The definition of MEMBER? is:

```
TO MEMBER? :ITEM :LIST
  IF :LIST = [] THEN OUTPUT "FALSE
  IF :ITEM = FIRST :LIST THEN OUTPUT "TRUE
  OUTPUT MEMBER? :ITEM BUTFIRST :LIST
END
```

In all LCSI versions use:

EMPTYP for EMPTY?
LISTP for LIST?
MEMBERP for MEMBER?
TYPE for PRINT1

There is also a primitive, EQUALP, which tests whether its two inputs are the same. Use this for comparing lists and words in place of =. (The equals sign works for lists on some LCSI versions, but not on others.)

The full IF syntax in LCSI LOGO is demonstrated by this example:

```
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.

Logo Poetry

In the previous instalment, we set you the problem of improving LOGO's poetry writing abilities. One possible method is to create a 'template' of a sentence structure (such as 'noun, verb, noun') and choose words from sublists of these parts of sentences. For example, using this template:

```
POEM2 [NOUN VERB NOUN]
```

we could have:

```
TO POEM2 :TEMPLATE
  IF EMPTY? :TEMPLATE PRINT "STOP
  POEM2.1 FIRST :TEMPLATE
  POEM2 BUTFIRST :TEMPLATE
END

TO POEM2.1 :WRD
  IF :WRD="NOUN (PRINT1"' ' GETRANDOM
  :NOUN)
  IF :WRD="VERB (PRINT2"' ' GETRANDOM
  :VERB)
END
```

This method will become very cumbersome if we introduce many more parts of speech. Let's take a closer look at variables again to see if we can improve on this. First of all, enter:

```
MAKE "ROSE "SWEET
MAKE "OTHERNAME "ROSE
```

Now you will find that:

```
PRINT :ROSE prints SWEET
PRINT :OTHERNAME prints ROSE
PRINT THING :OTHERNAME prints SWEET
```

THING gives us the value associated with a name. In the last case, the name following THING is the value of the variable OTHERNAME — in other words, ROSE. Using this idea, we can rewrite our poem procedure:

```
TO POEM2.1
  (PRINT1 "' ' GETRANDOM THING :WRD)
END
```

The procedure call POEM2.1 "NOUN assigns the value NOUN to the variable WRD. THING :WRD is then the value associated with NOUN — the list of nouns. Using the template:

```
POEM2 [ART ADJ NOUN ADV VERB PREP ART
  ADJ NOUN]
```

together with a suitable choice of vocabulary, we obtained the following output:

```
A GREEN PLANET LOUDLY SPUN UNDER A
  PARANOID DOME
A DILAPIDATED SHIP SLOWLY FLEW TOWARDS
  THE SPLENDID PLANET
```

Answers

1. For printing in reverse order:

```
TO PRINTR :LIST
  IF EMPTY? :LIST THEN PRINT " STOP
  PRINT1 LAST :LIST
  PRINT1 "' '
  PRINTR BUTLAST :LIST
END
```

The following procedure outputs the list in reverse order, rather than printing it.

```
TO REVERSE :LIST
  IF EMPTY? :LIST THEN OUTPUT []
  OUTPUT SENTENCE LAST :LIST REVERSE
  BUTLAST :LIST
END
```

2. We give a procedure to delete an item from a list in the main text of this instalment.