## 1994 Paper 1 Question 10

Modula-3 allows the user to declare arrays with any sort of contents — for instance arrays of integers, reals, `TEXT` or structures, but the index type for an array is restricted and in particular cannot be of type `TEXT`.

It is sometimes useful to achieve an effect analogous to having an array that can be indexed using values of type `TEXT`. One way of doing this is to use a structure known as a *hash table*: given an index value of type `TEXT` an integer is computed using a *hash function* and this is then used to index an array. The library procedure `Text.Hash` computes a suitable integer from a `TEXT` value. Two complications arise. First the integer computed by the hash function may lie outside the valid range of index values for the array. Secondly two different `TEXT` objects may give rise to the same hash value.

The problems can be resolved first by reducing the raw hash value modulo the size of the array and arranging that each array entry refers to the start of a linked list of (*index,value*) pairs. Retrieving a value from the table involves accessing the array to obtain the correct list of pairs and then scanning the list to find an index value that is identical (use the library function `Text.Equal`) with the `TEXT` index being sought. The corresponding value can then be returned. Storing into the table will involve adding a new (*index,value*) pair to one of the lists.

Design appropriate data structures for such a table, and write procedures to store and retrieve values, using the following signatures:

```
PROCEDURE Put(VAR table: Table; key, value: TEXT)
                                    RAISES {DuplicateKey}

PROCEDURE Get(READONLY table: Table; key: TEXT): TEXT
                                    RAISES {MissingKey}
```

[20 marks]