7.15
(a) subtype lc_letter is character range 'a'..'z';
upper : array (lc_letter) of character :=
('A', 'B', 'C', 'D', 'E', 'F',
'G', 'H', 'I', 'J', 'K', 'L',
'M', 'N', 'O', 'P', 'Q', 'R',
'S', 'T', 'U', 'V', 'W', 'X',
'Y', 'Z');

(b) subtype lc_letter is character range 'a'..'z';
function upper(l : lc_letter) return character is
    uc_offset : constant integer :=
        character'pos('A') - character'pos('a');
begin
    return character'val(character'pos(l) + uc_offset);
end upper;

7.21
3in
(a) r2 *:= 40             (b) r2 <<- 2
r3 <<- 2     r1 +:= r2
r1 +:= r3     r1 := *r1
r1 := *r1     r3 <<- 2
r1 := *r1
The left shifts effect multiplication by 4. The code sequence on the left is likely to be faster on
a modern machine, not because it is one instruction shorter, but because it performs only one
load instead of two.

7.24
double *a[n];  // array of n pointers to doubles
double (*b)[n];  // pointer to array of n doubles
double (*c[n])[n];  // array of n pointers to functions returning doubles
double (*d())[n];  // function returning pointer to array of n doubles

7.28
a) A program can easily keep a pointer to an object that it will never use again.
In fact it can keep pointers to an arbitrary number of such objects, if they are, say, linked
together in a list.
b) No. Consider the following:

foo := new object . . . if predicate() use foo

Any collector that will reclaim foo if and only if it is useless is capable of guessing the
results of an arbitrary predicate, which is equivalent to solving the (undecidable) halting
problem.

7.29

a) While this suggestion might indeed improve performance, it runs the risk of
creating dangling pointers. It might safely be used in conjunction with locks and keys.

b) Tenuring fits well with generational garbage collection: a tenured object can
be moved from the nursery to the oldest portion of the heap, where it will be considered
only if the collector runs out of memory.

7.30

If the reference count in a tombstone goes to zero but the object to which it refers
has not been reclaimed, then it will never be possible to reclaim it, and the program has leaked
storage. If there are circular references, however, and no external reference into a cycle, then all
tombstones in the cycle will have nonzero reference counts, and we will fail to detect the leak.