```
enum gray_values {
  BLACK = 0,
  DARK_GRAY = 64,
  GRAY = 128,
  LIGHT_GRAY = 192,
};
```

Is this practice legal?

A: C99 This practice is indeed legal in C99 (and is supported by some pre-C99 compilers as well). Allowing a "trailing comma" makes enumerations easier to modify, because we can add a constant to the end of an enumeration without changing existing lines of code. For example, we might want to add `WHITE` to our enumeration:

```
enum gray_values {
  BLACK = 0,
  DARK_GRAY = 64,
  GRAY = 128,
  LIGHT_GRAY = 192,
  WHITE = 255,
};
```

The comma after the definition of `LIGHT_GRAY` makes it easy to add `WHITE` to the end of the list.

One reason for this change is that C89 allows trailing commas in initializers, so it seemed inconsistent not to allow the same flexibility in enumerations. Incidentally, C99 also allows trailing commas in compound literals. C99

Q: Can the values of an enumerated type be used as subscripts?

A: C99 Yes, indeed. They are integers and have—by default—values that start at 0 and count upward, so they make great subscripts. In C99, moreover, enumeration constants can be used as subscripts in designated initializers. Here's an example:

```
enum weekdays {MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY};
const char *daily_specials[] = {
  [MONDAY] = "Beef ravioli",
  [TUESDAY] = "BLTs",
  [WEDNESDAY] = "Pizza",
  [THURSDAY] = "Chicken fajitas",
  [FRIDAY] = "Macaroni and cheese"
};
```

Exercises

Section 16.1

1. In the following declarations, the `x` and `y` structures have members named `x` and `y`:

```
struct { int x, y; } x;
struct { int x, y; } y;
```

Are these declarations legal on an individual basis? Could both declarations appear as shown in a program? Justify your answer.