A: Yes, there is. Some programmers use the following idiom when calling `malloc` to allocate memory for a single object:

```
p = malloc(sizeof(*p));
```

Since `sizeof(*p)` is the size of the object to which `p` will point, this statement guarantees that the correct amount of memory will be allocated. At first glance, this idiom looks fishy: it's likely that `p` is uninitialized, making the value of `*p` undefined. However, `sizeof` doesn't evaluate `*p`, it merely computes its size, so the idiom works even if `p` is uninitialized or contains a null pointer.

To allocate memory for an array with `n` elements, we can use a slightly modified version of the idiom:

```
p = malloc(n * sizeof(*p));
```

**Q: Why isn't the `qsort` function simply named `sort`? [p. 440]**

A: The name `qsort` comes from the Quicksort algorithm published by C. A. R. Hoare in 1962 (and discussed in Section 9.6). Ironically, the C standard doesn't require that `qsort` use the Quicksort algorithm, although many versions of `qsort` do.

**Q: Isn't it necessary to cast `qsort`'s first argument to type `void *`, as in the following example? [p. 441]**

```
qsort((void *) inventory, num_parts, sizeof(struct part),
      compare_parts);
```

A: No. A pointer of any type can be converted to `void *` automatically.

***Q: I want to use `qsort` to sort an array of integers, but I'm having trouble writing a comparison function. What's the secret?**

A: Here's a version that works:

```
int compare_ints(const void *p, const void *q)
{
  return *(int *)p - *(int *)q;
}
```

Bizarre, eh? The expression `(int *)p` casts `p` to type `int *`, so `*(int *)p` would be the integer that `p` points to. A word of warning, though: Subtracting two integers may cause overflow. If the integers being sorted are completely arbitrary, it's safer to use `if` statements to compare `*(int *)p` with `*(int *)q`.

***Q: I needed to sort an array of strings, so I figured I'd just use `strcmp` as the comparison function. When I passed it to `qsort`, however, the compiler gave me a warning. I tried to fix the problem by embedding `strcmp` in a comparison function:**