```
int compare_strings(const void *p, const void *q)
{
  return strcmp(p, q);
}
```

Now my program compiles, but `qsort` doesn't seem to sort the array. What am I doing wrong?

A: First, you can't pass `strcmp` itself to `qsort`, since `qsort` requires a comparison function with two `const void *` parameters. Your `compare_strings` function doesn't work because it incorrectly assumes that `p` and `q` are strings (`char *` pointers). In fact, `p` and `q` point to array elements containing `char *` pointers. To fix `compare_strings`, we'll cast `p` and `q` to type `char **`, then use the `*` operator to remove one level of indirection:

```
int compare_strings(const void *p, const void *q)
{
  return strcmp(*(char **)p, *(char **)q);
}
```

Exercises

Section 17.1

1. Having to check the return value of `malloc` (or any other memory allocation function) each time we call it can be an annoyance. Write a function named `my_malloc` that serves as a "wrapper" for `malloc`. When we call `my_malloc` and ask it to allocate `n` bytes, it in turn calls `malloc`, tests to make sure that `malloc` doesn't return a null pointer, and then returns the pointer from `malloc`. Have `my_malloc` print an error message and terminate the program if `malloc` returns a null pointer.

Section 17.2

W 2. Write a function named `duplicate` that uses dynamic storage allocation to create a copy of a string. For example, the call

```
p = duplicate(str);
```

would allocate space for a string of the same length as `str`, copy the contents of `str` into the new string, and return a pointer to it. Have `duplicate` return a null pointer if the memory allocation fails.

Section 17.3

3. Write the following function:

```
int *create_array(int n, int initial_value);
```

The function should return a pointer to a dynamically allocated `int` array with `n` members, each of which is initialized to `initial_value`. The return value should be `NULL` if the array can't be allocated.

Section 17.5

4. Suppose that the following declarations are in effect:

```
struct point { int x, y; };
struct rectangle { struct point upper_left, lower_right; };
struct rectangle *p;
```