"normal" parameter, so you can't remove the parameter n. Instead, assume that it represents one of the numbers to be compared.

Ⓦ 2. Write a simplified version of `printf` in which the only conversion specification is `%d`, and all arguments after the first are assumed to have `int` type. If the function encounters a `%` character that's not immediately followed by a `d` character, it should ignore both characters. The function should use calls of `putchar` to produce all output. You may assume that the format string doesn't contain escape sequences.

3. Extend the function of Exercise 2 so that it allows two conversion specifications: `%d` and `%s`. Each `%d` in the format string indicates an `int` argument, and each `%s` indicates a `char *` (string) argument.

4. Write a function named `display` that takes any number of arguments. The first argument must be an integer. The remaining arguments will be strings. The first argument specifies how many strings the call contains. The function will print the strings on a single line, with adjacent strings separated by one space. For example, the call

```
display(4, "Special", "Agent", "Dale", "Cooper");
```

will produce the following output:

```
Special Agent Dale Cooper
```

5. Write the following function:

```
char *vstrcat(const char *first, ...);
```

All arguments of `vstrcat` are assumed to be strings, except for the last argument, which must be a null pointer (cast to `char *` type). The function returns a pointer to a dynamically allocated string containing the concatenation of the arguments. `vstrcat` should return a null pointer if not enough memory is available. *Hint:* Have `vstrcat` go through the arguments twice: once to determine the amount of memory required for the returned string and once to copy the arguments into the string.

6. Write the following function:

```
char *max_pair(int num_pairs, ...);
```

The arguments of `max_pair` are assumed to be "pairs" of integers and strings; the value of `num_pairs` indicates how many pairs will follow. (A pair consists of an `int` argument followed by a `char *` argument). The function searches the integers to find the largest one; it then returns the string argument that follows it. Consider the following call:

```
max_pair(5, 180, "Seinfeld", 180, "I Love Lucy",
            39, "The Honeymooners", 210, "All in the Family",
            86, "The Sopranos")
```

The largest `int` argument is 210, so the function returns `"All in the Family"`, which follows it in the argument list.

Section 26.2

Ⓦ 7. Explain the meaning of the following statement, assuming that `value` is a variable of type `long int` and `p` is a variable of type `char *`:

```
value = strtol(p, &p, 10);
```

8. Write a statement that randomly assigns one of the numbers 7, 11, 15, or 19 to the variable `n`.

Ⓦ 9. Write a function that returns a random `double` value *d* in the range 0.0 ≤ *d* < 1.0.

10. Convert the following calls of `atoi`, `atol`, and `atoll` into calls of `strtol`, `strtol`, and `strtoll`, respectively.