```
  if (cur == NULL)
    return list;                    /* n was not found */
  if (prev == NULL)
    list = list->next;              /* n is in the first node */
  else
    prev->next = cur->next;         /* n is in some other node */
  free(cur);
  return list;
}
```

Deleting the first node in the list is a special case. The `prev == NULL` test checks for this case, which requires a different bypass step.

Ordered Lists

When the nodes of a list are kept in order—sorted by the data stored inside the nodes—we say that the list is ***ordered.*** Inserting a node into an ordered list is more difficult (the node won't always be put at the beginning of the list), but searching is faster (we can stop looking after reaching the point at which the desired node would have been located). The following program illustrates both the increased difficulty of inserting a node and the faster search.

PROGRAM

Maintaining a Parts Database (Revisited)

Let's redo the parts database program of Section 16.3, this time storing the database in a linked list. Using a linked list instead of an array has two major advantages: (1) We don't need to put a preset limit on the size of the database; it can grow until there's no more memory to store parts. (2) We can easily keep the database sorted by part number—when a new part is added to the database, we simply insert it in its proper place in the list. In the original program, the database wasn't sorted.

In the new program, the `part` structure will contain an additional member (a pointer to the next node in the linked list), and the variable `inventory` will be a pointer to the first node in the list:

```
struct part {
  int number;
  char name[NAME_LEN+1];
  int on_hand;
  struct part *next;
};

struct part *inventory = NULL;    /* points to first part */
```

Most of the functions in the new program will closely resemble their counterparts in the original program. The `find_part` and `insert` functions will be more complex, however, since we'll keep the nodes in the `inventory` list sorted by part number.