```
struct part *find_part(int number)
{
  struct part *p;

  for (p = inventory;
       p != NULL && number > p->number;
       p = p->next)
    ;
  if (p != NULL && number == p->number)
    return p;
  return NULL;
}

/**********************************************
 * insert: Prompts the user for information about a new   *
 *         part and then inserts the part into the        *
 *         inventory list; the list remains sorted by     *
 *         part number. Prints an error message and       *
 *         returns prematurely if the part already exists *
 *         or space could not be allocated for the part.  *
 **********************************************/
void insert(void)
{
  struct part *cur, *prev, *new_node;

  new_node = malloc(sizeof(struct part));
  if (new_node == NULL) {
    printf("Database is full; can't add more parts.\n");
    return;
  }

  printf("Enter part number: ");
  scanf("%d", &new_node->number);

  for (cur = inventory, prev = NULL;
       cur != NULL && new_node->number > cur->number;
       prev = cur, cur = cur->next)
    ;
  if (cur != NULL && new_node->number == cur->number) {
    printf("Part already exists.\n");
    free(new_node);
    return;
  }

  printf("Enter part name: ");
  read_line(new_node->name, NAME_LEN);
  printf("Enter quantity on hand: ");
  scanf("%d", &new_node->on_hand);

  new_node->next = cur;
  if (prev == NULL)
    inventory = new_node;
  else
    prev->next = new_node;
}
```