Unit 15: Dynamic Data Structures

Dynamically allocated data structures play a big role in C programming, since they can be linked together to form lists, trees, and other highly flexible data structures. The most fundamental data structure is a linked list, which we shall focus on in this unit: each element of a list is represented by a structure which contains data and/or pointers to other data. The elements of the list are "linked" by a pointer from one element to the next element.
The relevant Book Sections for this unit are:

- Section 17.5: Linked lists
- Section 17.6: Pointers to pointers (e.g. required when modifying the very first element of a list and updating the pointer to that list. See example `add_to_list` on pages 438-439)

See example inventory2.c which is a rewritten version of the inventory.c program we started with, but now being used with a list of parts (instead of an array of parts).

Project u15: Removing a List Element

Modify the provided inventory2.c program of Section 17.5 by adding an `r` (remove) command that allows the user to remove a part from the database. The part to be removed is selected by part number. Ensure that the program also handles the case that the list becomes empty.

```
% ./u15_inv2remove
Enter operation code: i
Enter part number: 1
Enter part name: TFT 24" Monitor
Enter quantity on hand: 2

Enter operation code: i
Enter part number: 2
Enter part name: Graphics Card
Enter quantity on hand: 1
Enter price: 414.90

Enter operation code: i
Enter part number: 0
Enter part name: Memory 4GB
Enter quantity on hand: 1

Enter operation code: p

Part Number   Part Name                      Quantity on Hand
      0       Memory 4GB                            1
```

```
      1       TFT 24" Monitor                    2
      2       Graphics Card                      1

Enter operation code: r
Enter part number: 1
Enter operation code: p

Part Number   Part Name                  Quantity on Hand
      0       Memory 4GB                         1
      2       Graphics Card                      1

Enter operation code: q

%
```

Refer to the provided test-cases and compare the output of your program with the provided output of the test-cases. Your program should behave exactly like the test-cases before you hand it in. Hand in your program as `u15_inv2remove.c`.