Unit 13: Structures and Enumerations

A structure is a collection of values (members), possibly of different types. An enumeration is an integer type whose values are named by the programmer. The following Book Sections are relevant for this unit:

- Section 16.1 shows how to declare structure variables and perform basic operations on them
- Section 16.2 explains how to define structure types, which allows to write functions that accept structure arguments or return structures.
- Section 16.3 shows how arrays and structures can be nested.
- Section 16.5 shows how enumeratios are used

The running example is a Parts Database. This program is provided as inventory.c – and we shall extend this program through a number of programming projects. It uses structures, enumerations, and functions to perform operations on arrays of structures.

Project u13: Inventory Modification 1

The provided inventory program consists of 7 functions: the main function, four auxiliary functions for each operation: insert, search, update, print, and the functions `read_line` which is provided in the separate files `readline.h` and `readline.c`.
Perform the following modification to the inventory.c program:

- Add a `price` member of type float to the part structure.
- The `insert` function should ask the user for the price of a new item, posing as last question: "Enter price: ".
- The search function should display the price preceded by "Price:" followed by the price as float number with two digits after the comma. The price has to be printed after the quantity at hand in a new line.
- Extend the print function so that the header is printed in a new line and is extended with "Price", preceeded by 2 spaces following "Quantity on Hand". The price value has to be aligned with "Price" with 4 digits before the comma (filled up with spaces), and two digits following the comma.
- Add a new command 'c' that allows to change the price: add a function `update_price` which works like the update function, but asks instead of the quantity at hand for "Enter new price :" and assigns the new price to the price member of the selected part.

Example:

```
% ./u13_invmod1
Enter operation code: i
Enter part number: 1
Enter part name: TFT 24" Monitor
Enter quantity on hand: 2
Enter price: 1810.00

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: p

Part Number   Part Name                    Quantity on Hand  Price
      1       TFT 24" Monitor                     2          1810.00
      2       Graphics Card                       1           414.90

Enter operation code: c
Enter part number: 1
Enter new price: 1450.00

Enter operation code: p

Part Number   Part Name                    Quantity on Hand  Price
      1       TFT 24" Monitor                     2          1450.00
      2       Graphics Card                       1           414.90

Enter operation code: q

%
```

Also refer to the provided test-cases for the exact behaviour of the price-extension. Your program should behave exactly like the test-cases before you hand it in. Note, no other changes should be applied to the program. Hand in the modified program as `u13_invmod1.c` (you do not need to hand in `read_line.c`)