Unit 16: Pointers to Functions

Although it might seem that pointers to functions aren't relevant to the average programmer, that couldn't be further from the truth. In fact, some of the most useful functions in the C library require a function pointer as argument. One of these is the function qsort, which belongs to the `<stdlib.h>` header. The function qsort is a general purpose sorting function that is capable of sorting any array, based on any criteria that we choose. Since the elements of the array that it sorts may be of any type - even a structure - qsort must be told how to determine which of the two array elements is "smaller". We'll provide this information to qsort by providing a "comparison function".
Relevant for this unit are the Book Sections:

- Section 17.7: Function Pointers

Project u16: Sorting the Inventory, Modification 3

Extend the inventory program (inventory.c) so that the program allows to sort the inventory list by part number, name, or quantity. Use the qsort function (see Book Section 17.7, qsort) of the standard C library. Write the proper compare-functions for comparing the parts. Extend the inventory program with a sort command, 'o' (for Order, since the more intuitive 's' is already used for "search"). The 'o' command then asks for a subcommand and which property to sort by. Make sure that the sort subcommands are handled differently from the other commands. When entering the subcommand mode prompt the user with "Sort for property [pnqx]: ". The user can then enter one of the following subcommands and the program prints a short message about the sort mode:

- subcommand 'p' for part number and prints: "Sorting by part number.", followed by a newline. (sorted from lowest to highest number)
- subcommand 'n' for name and prints: "Sorting by name.", followed by a newline. (sorted in ascending order)
- subcommand 'q' for quantity and prints: "Sorting by quantity.", followed by a newline. (sorted from highest quantity to lowest (!))
- subcommand 'x' exiting the submenu without sorting and prints "Not Sorting.", followed by a newline. The program continues with prompting for a command.

In case a wrong subcommand is entered the program prompts the user again for a subcommand.
Example:

```
% ./u16_invmod3
Enter operation code: i
Enter part number: 1
Enter part name: PartB
Enter quantity on hand: 30
```

```
Enter operation code: i
Enter part number: 2
Enter part name: PartC
Enter quantity on hand: 20

Enter operation code: i
Enter part number: 3
Enter part name: PartA
Enter quantity on hand: 25

Enter operation code: n
Illegal code

Enter operation code: o
Sort for property [pnqx]: n
Sorting by name.

Enter operation code: p

Part Number   Part Name                  Quantity on Hand
      3       PartA                             25
      1       PartB                             30
      2       PartC                             20

Enter operation code: o
Sort for property [pnqx]: q
Sorting by quantity.

Enter operation code: p

Part Number   Part Name                  Quantity on Hand
      1       PartB                             30
      3       PartA                             25
      2       PartC                             20

Enter operation code: q

%
```

Refer to the test-cases to see examples of a proper use of the new sort feature. Make sure your program produces identical output as the provided test-cases before handing in your solution. Hand in your program as `u16_invmod3.c`.