9.3 *(Querying an Array of `Invoice` Objects)* Use the class `Invoice` provided in the `ex09_03` folder with this chapter's examples to create an array of `Invoice` objects. Use the sample data shown in Fig. 9.8. Class `Invoice` includes four properties—a `PartNumber` (type `int`), a `PartDescription` (type `string`), a `Quantity` of the item being purchased (type `int`) and a `Price` (type `decimal`). Perform the following queries on the array of `Invoice` objects and display the results:

a) Use LINQ to sort the `Invoice` objects by `PartDescription`.
b) Use LINQ to sort the `Invoice` objects by `Price`.
c) Use LINQ to select the `PartDescription` and `Quantity` and sort the results by `Quantity`.
d) Use LINQ to select from each `Invoice` the `PartDescription` and the value of the `Invoice` (i.e., `Quantity * Price`). Name the calculated column `InvoiceTotal`. Order the results by `Invoice` value. [*Hint:* Use `let` to store the result of `Quantity * Price` in a new range variable `total`.]
e) Using the results of the LINQ query in part (d), select the `InvoiceTotals` in the range $200 to $500.

Part number	Part description	Quantity	Price
83	Electric sander	7	57.98
24	Power saw	18	99.99
7	Sledge hammer	11	21.50
77	Hammer	76	11.99
39	Lawn mower	3	79.50
68	Screwdriver	106	6.99
56	Jig saw	21	11.00
3	Wrench	34	7.50

Fig. 9.8 | Sample data for Exercise 9.3.

9.5 *(Sorting Pets Names by Length)* Write a console app that inputs pets' names into a `List<string>`. Perform the following queries on the `List` and display your results:

a) Use LINQ to sort the `List` in ascending order on the basis of number of characters in the name.

b) Use LINQ to sort the `List` in descending order on the same basis.

c) Display the `List` count after removing duplicate names from the list.