You are developing classes for the products books and software, which must be processed slightly differently.
Potentially, in the future there will be many other product types added to this hierarchy, and you wish to provide a general class `Product` with `Book` and `Software` as sub-classes.

1. Write a class for a `Product`
   - Each `Product` has a data member which holds the net price, and a constructor which sets this price.
   - Each `Product` has a method `getGrossPrice(),` which calculates and returns the gross price (the gross price includes VAT at 21%)

2. Write 2 classes which inherit from the general `Product` class, of type `Software` and `Book`
   - The gross price for Software includes VAT at 21%,
   - Books are free of VAT, so the gross price is unchanged from the net price, and you will need to re-define the `getGrossPrice` method in this class

3. Write a program which does the following:

   a. Declare an array of 10 pointers to `Product`

   b. Declare a pointer to a `Book` and a pointer to `Software`

   c. Ask the user to enter details of the book, and of the software item, create the two items dynamically and store their addresses in your pointers.

   d. Check your method `getGrossPrice` works correctly with each type, and then add them as the first 2 pointers in the array of product.

   e. Now ask the user to enter details of the remaining 8 items to be stored in your array. For each they must give the net price and say whether it is a book or software. Enter each item with
   `a[i] = new Software(price) or a[i] = new Book(price)`

   f. Cycle through the array and print the gross price of each item using the `getGrossPrice` method. *Do your first two items return the same gross price as they did when you checked in an earlier step? Should they?*

   g. Sort the array in ascending order of gross price, using an efficient sorting mechanism

   h. Repeat step f

NB: We are going to use this example again later using polymorphism.