TU Dublin - Tallaght Campus
Department of Computing
Software Development 2
CA2 (40%)

This is an <u>individual</u> assignment.
Please note rules regarding plagiarism

Due 6pm Monday 20th April via upload to Moodle

<u>Instructions & Deliverables:</u>
Upload a **.py file solution** to the problem **and** a **set of test tables**.

Also, you **<u>must include</u>** a cover sheet (a single page) including your name, student number, module name, i.e. Software Development 2, and declare that the work submitted is entirely your own work.

Place all three in a folder and upload the zipped folder.
Include your name and student number as a comment at the start of your program.

Marks will be awarded for successful compilation and a working program according to the specification, good program design, style, comments and rigorous testing.

<u>Problem:</u>

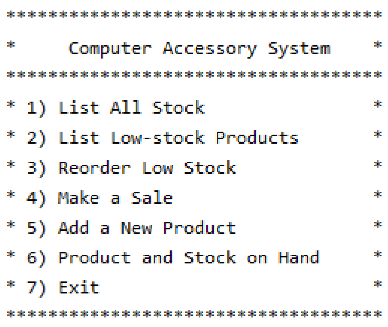
Write a Python program for a Computer Accessories Sales System.

Download the **products.csv** file from Moodle into your project folder in order to use it in your system. This is a **csv** file where each line represents a product's details: in each row:
the first value is the product ID,
the second value is the product name
the third value is the product cost price
the fourth value is the product selling price
the fifth value is the product quantity on hand
the sixth value is the product re-order level

For your system:
Create a **static** class **ProductUtility** :

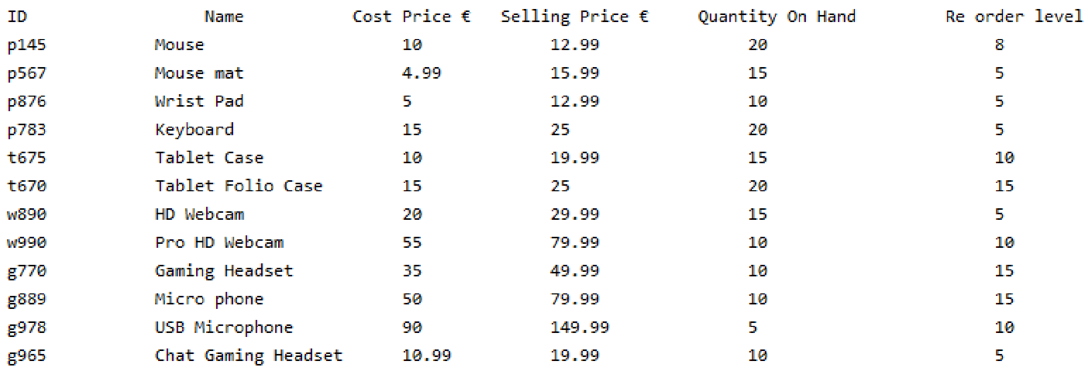
with a **class** variable, VAT with a value of 23, which represents the 23% VAT rate, which will be charged on the products selling price.
Code the following **<u>static</u>** methods in the ProductUtility class:

A menu() method that displays a menu as follows:



```
************************************
*     Computer Accessory System    *
************************************
* 1) List All Stock                *
* 2) List Low-stock Products       *
* 3) Reorder Low Stock             *
* 4) Make a Sale                   *
* 5) Add a New Product             *
* 6) Product and Stock on Hand     *
* 7) Exit                          *
************************************
```

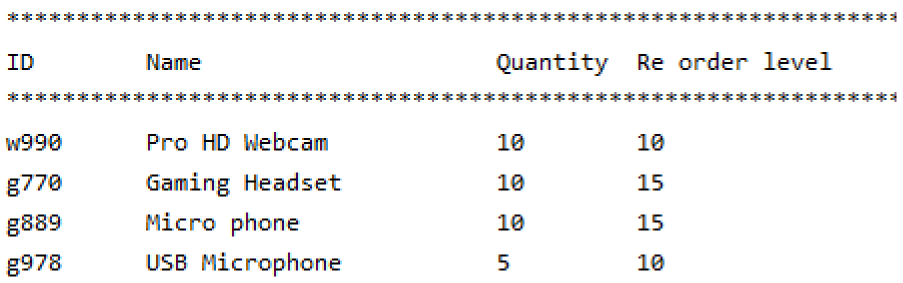
A **display_stock() static** method that takes the 2D list of stock items as a parameter and displays the stock items in a formatted manner as follows:



```
ID              Name          Cost Price €  Selling Price €    Quantity On Hand         Re order level
p145          Mouse                 10           12.99             20                       8
p567          Mouse mat             4.99         15.99             15                       5
p876          Wrist Pad             5            12.99             10                       5
p783          Keyboard              15           25                20                       5
t675          Tablet Case           10           19.99             15                       10
t670          Tablet Folio Case     15           25                20                       15
w890          HD Webcam             20           29.99             15                       5
w990          Pro HD Webcam         55           79.99             10                       10
g770          Gaming Headset        35           49.99             10                       15
g889          Micro phone           50           79.99             10                       15
g978          USB Microphone        90           149.99            5                        10
g965          Chat Gaming Headset   10.99        19.99             10                       5
```

If there is no stock in the 2D list your method should print an appropriate message, otherwise display the stock in a manner similar to above.

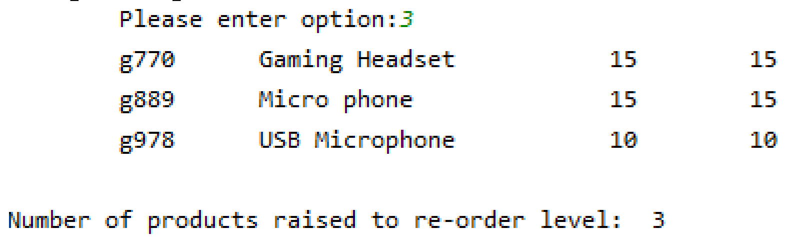
A **display_low_stock() static** method that takes the 2D list of stock items as a parameter and displays the details of any stock item where the quantity on hand is less than or equal to the re-order level, as below. The method should also keep a count of the number of products that have quantity on hand less than or equal to the reorder level. This figure should also be printed to the screen after the display of products.



```
*****************************************************************
ID        Name                     Quantity  Re order level
*****************************************************************
w990      Pro HD Webcam            10        10
g770      Gaming Headset           10        15
g889      Micro phone              10        15
g978      USB Microphone           5         10
```

A **reorder_low_stock() static** method that takes as a parameter the 2D list of stock items, and for any product where the quantity is below the re-order level, the quantity should be raised to the reorder level, thereby simulating the reordering of stock. The updated product details should be printed similar to below. A count should also be kept of the number of products re-ordered and this count printed to the screen.

Sample Output:

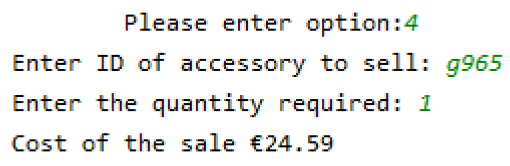


```
Please enter option:3
g770      Gaming Headset          15        15
g889      Micro phone             15        15
g978      USB Microphone          10        10

Number of products raised to re-order level:  3
```

A **make_sale() static** method that simulates making a sale of a product. This method takes the 2D list of stock items, the id of the stock item being sold and the number of that item required as parameters. The method should first check that sufficient quantity of the required stock item is available to make the sale, you may code another **static** method, get_qty() to determine this and return the quantity on hand of the required stock item from this method. If there is sufficient quantity on hand to fill the sale, calculate and return the cost of this sale from the **make_sale()** method, note that the selling price in the stock list does not include VAT, which is a tax that is add to the selling price, so VAT should be added to the selling price. Also, the quantity on hand should be updated to reflect the number of items sold. If the sale cannot be made a cost of **0** should be returned from the method. Print the cost of the sale as below.
The user inputs in the sample output below should be inputted in the **main** body of the code and passed to the method, see **option 4** below:

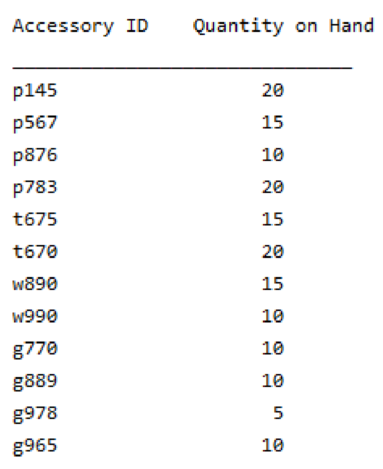
Sample Output:



```
    Please enter option:4
Enter ID of accessory to sell: g965
Enter the quantity required: 1
Cost of the sale €24.59
```

A **static method add_product()** which takes as parameters the 2D stock list and values for all the product's details and adds that new product to the 2D list of stock items.

The most frequent request from users of the system is to generate a list of product IDs and their corresponding quantity on hand. In order to generate faster access to these values, write a **static create_dictionary**() **method**, which takes as a parameter the 2D list of stock items: in the method create a dictionary with the product ids as the keys and their corresponding quantity on hand as their values.
Then, in the method, display the dictionary key-value pairs as shown below:
Sample Output



```
Accessory ID    Quantity on Hand
______________________________
p145                 20
p567                 15
p876                 10
p783                 20
t675                 15
t670                 20
w890                 15
w990                 10
g770                 10
g889                 10
g978                  5
g965                 10
```

In the main body of the program, implement the following logic:

The csv file should be read into the main body of your code as a 2D list, see Files Notes on Moodle and don't forget to **import csv**.

The program should then implement a menu driven system with the menu being displayed and the user entering the options required until the Exit option, 7, is selected.

Invalid options should be caught.

If **option 1** is selected:
The 2D list of stock items should be passed to the display_stock() method and the method executed:

If **option 2** is selected:
The display_low_stock() method should be called with the 2D list of stock items passed in as an argument.

If **option 3** is selected:
The reorder_low_stock() method should be called with the 2D list of stock items passed to it as an argument.

If **option 4** is selected:
The user should be prompted to enter the ID of the product that they wish to sell, they should also be prompted to enter the number of that item that they wish to sell. These values should be passed into the make_sale() static method with the 2D list of stock items, the method then returns the cost of the sale. The return value from the make_sale() method should be evaluated and if it is 0 an appropriate message printed to the screen and otherwise print the cost of the sale to the screen.

If **option 5** is selected:

The user wishes to add a new product to the 2D list of stock items. The user should be prompted for all the values of a product, i.e. ID, name, cost price, which should be greater than 0, selling price, which should be greater than the cost price, quantity on hand and re-order level. These values and the 2D stock list should be passed to the add_product() static method, which adds the product to the stock list.

If **option 6** is selected:
Call the create_dictionary() method, passing the 2D stock list to it.

After the user has selected the exit option to exit the menu system, the program should write the 2D list of stock items to a csv file called **stock.csv**.