Q1) Write a class that encapsulates data for a sales person. Class **`SalesPerson`** has an ID number (`int`), a name (`string`), and a list of sales (`float each`). It provides a constructor `__init__` that initializes ID and name, and initializes the list of sales to an empty list. Member methods for `SalesPerson` include:

- getter methods for ID and name.
- setter method for name
- `enterSale(aSale)` appends the parameter value into the list of sales.
- `totalSales()` returns Salesperson's sum of total sales.
- `getSales ()` returns a list of all sales
- `metQuota(quota)`, a Boolean method that returns `True` if the total sales meet or exceed the `quota` provided, and `False` if not.
- `compareTo (otherPerson)` that compares this Salesperson (`self`) to another (provided as a parameter) based on the values of their total sales amounts.
- a `__str__()` method that displays the ID, name, and total sales.

Test `SalesPerson` thoroughly before continuing.

Q2) Write a class to keep track of the sales made by different salespersons. Class **`SalesForce`** maintains a list of `SalesPerson` objects (created in Q01). The constructor has no parameters and should initialize the list. Member methods for `SalesForce` include:

- `addData(fileName)` accepts a file name as its only parameter. It inputs information for all salespersons from that file. Each line of the file will contain data in the form:
  ID name Sales amounts.
  Note that each line contains the id, name and sales amounts for one Salesperson. Each Salesperson may have 0 or more sales.
- `quotaReport (quota)` prints each salesperson's ID, name and total sales and whether or not the salesperson met the sales quota.
- `topSalesPerson ()` returns the Salesperson object representing the individual with the highest sales amount.

- `individualSales (id)` prints the sales record (all of his sales) and the total of his sales for the employee with the ID specified. If there is no employee with the given ID, the method prints an appropriate message.

Provided is **`SalesTestDriver.py`**, which contains a main method that inputs the name of the file from which the sales information is to be read, instantiates an object of type `SalesForce`, invokes the `SaleForce getData()` method with the file name, invokes the `SaleForce quotaReport()` method with a quota amount, invokes the `SaleForce topSalesPerson()` method and invokes the `SalesForce individualSales()` method a couple of times. Test data are provided in the file `salesdata.txt`.

Submission:
Submit `SalesPerson.py` and `SalesForce.py` in the usual manner by the date and time shown on Oaks.

Policies, documentation, formatting, collaboration:
The policies stated in the class syllabus are in effect for this and all assignments.