**Flower Ordering System**

A flower ordering system automates the process of customers ordering flowers. Different types of flowers are available. A customer can place an order for a specific quantity of a single type of flower, and the total price will be calculated based on the flower type and the number of stems ordered.

**Files Provided:**

* **flowers.txt:** Contains flower details like flower type and price per stem in the following format:
  + flower\_id#flower\_name#price\_per\_stem
  + Sample records are:
  + 1#Rose#2.50
  + 2#Lily#3.00
  + 3#Tulip#2.00
  + 4#Orchid#5.00
* **orders.txt:** Contains order details like order ID, customer name, flower name, quantity, and total price in the following format:
  + order\_id#customer\_name#flower\_name#quantity#total\_price
  + Sample records are:

F101#Alice#Rose#12#30.00

**Tasks:**

1. **auto\_generate\_orderid() Function:**
   * Consider the file orders.txt is always present.
   * Auto-generate the order ID with the following rules and return the generated order ID:
     + Order ID is a unique number, starting from '101'.
     + It is given to every new order, as "F101", "F102", and so on.
     + If the file has the last record as, for example, F102#Shiv#Lily#20#60.00, then the order ID should be generated as '103' and returned.
     + If no previous record is present in the file, then the order ID should be generated as '101' and returned.
2. **place\_order() Function:**
   * It should accept three arguments in the given order: customer\_name, flower\_name, quantity.
   * Then validate the arguments as given below.
   * **Validation rules for customer\_name:**
     + It should be only an alphabetic string of 3 to 15 characters whose first character must be in uppercase, and the rest of the characters must be lowercase. Example: 'Newton', 'Einstein', 'Hawking', 'Mary' are some of the valid names.
     + If the above validations are met, proceed to the next validation; else, return with a status code 4.
   * **flower\_name should be present in the file flowers.txt.**
     + Do a case-insensitive search.
     + If it is present, then proceed to the next step; else, return with a status code 3.
   * Invoke auto\_generate\_orderid() function to generate a new order\_id.
   * Get the price\_per\_stem for the flower\_name ordered from the flowers.txt file.
   * Calculate the total price by multiplying the quantity and price\_per\_stem (use bc for floating-point calculations).
   * Now, insert the following data into the file orders.txt in the format (as mentioned in the "Files Provided" section):
     + order\_id (Add the letter 'F' at its beginning)
     + customer\_name
     + flower\_name
     + quantity
     + total\_price
   * Return with a status code 5.
   * Assume: Unlimited flowers are available.
   * **Note:**
     + The order of the file contents should not be changed.
     + Perform case-insensitive matching for flower\_name.

**File Structures:**

**flowers.txt:**

1#Rose#2.50

2#Lily#3.00

3#Tulip#2.00

4#Orchid#5.00

**orders.txt:**

F101#Alice#Rose#12#30.00

F102#Bob#Lily#10#30.00