Extra Lab Tasks

1. Design and implement an OOP in C++ to keep inventory of a florist. The florist wants to store information on all flowers in the shop.

- For each flower, she needs to keep track of the flower `name`, `quantity on hand`, and `price`. Each flower must be given a unique `id`, that is generated sequentially starting from 1. Additionally, she wants to know the `total number of flowers` in the shop.

Example: If the florist has only the following flowers, `total number of flowers`=22

```
Flower id=1, Name= "Daisy", Number on Hand=12, Price=2.5
Flower id=2, Name= "Cactus", Number on Hand=7, Price=0.5
Flower id=3, Name= "Rose", Number on Hand=3, Price=5.0
```

- Your class definition should contain a buy and a sell function in addition to the other methods, constructor and destructor.
    - Buy function accepts a parameter *quantity* and is used to buy more flowers and increases the `number on hand` value by adding the *quantity* value to it.
    - Sell function is used to sell flowers to customers and decreases the `number on hand` value by subtracting *quantity* from it.
    - Note that when you sell and buy, the total number of flowers also change appropriately
- All functions that do NOT modify the data members must be declared as const.

- Use the following menu as a guideline for your main function.

Please enter your choice

1. Create a new flower
2. Buy flower
3. Sell flower
4. Modify flower
5. Exit

- Display the menu until the user enters 5 or 6.
- Create a new flower option is used to create a flower (*dynamic object*)
- Buy flower is used to buy more flowers for an existing flower.
- Sell flower is used to sell flowers to customers for existing flowers.
- Modify a flower is used to change the name or price of an existing flower.
- Exits terminates the program and all dynamic objects are deleted
- Exit without saving exits without writing anything into the flowers.dat file.

2. Use the UML diagram given below to create the class

Customer
custId:int name:string custCount:static int
Customer( ) Customer(string) Customer(const Customer&) ~Customer( ) int getCustId( ) string getName( ) static int getCustCount( ) void setName(sting)

Instructions:

- The Default Constructor prompts user to enter data from keyboard for the private data member (name)
- The parameterized constructor accepts customer name as parameter
- The copy constructor copies the value(s) of an existing object to the new object

Whenever a new object is created, the compiler assigns unique customer id for each customer.

- Destructor decrements the custCount by one and prints the name of the customer that has been destroyed. (i.e: "Customer XXXX has been deleted!")
- Getter methods should return the corresponding data member.
- Setter methods should receive parameter from main() and change the corresponding data member.
- Initialize the static data member custCount to 0.

a) Create customer class

b) Write a main( ) function that will include the following steps.

<customer.cpp>

- Include necessary header files,
- Create a dynamic customer object,
- Display the id and the name of the customer object,
- Create dynamic array object for 20 customers,
- Display the id, and name of all customers in the array,
- Change the name of the customers in the 3rd and 5th locations (assign your own name and your best friend's name)
- Display the id and the name of the 3rd and the 5th customer objects,
- Display the total number of customers,
- Delete all objects that you have created.