NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES
ISLAMABAD CAMPUS
CS103 - Computer Programming(All sections)

Spring 2014 Assignment#04
Issue Date: February 25^{th}, 2014 Due Date: **March 2^{nd}, 2014 @ 11:30pm**

Instructions

- Submit the source code of the assignment via Slate only.
- Use proper naming convention to name the file containing the source code. For example, the file containing the source code for second question of the third assginment should be named as A3Q2.cpp, that of the third question of the second assignment as A2Q3.cpp, etc.
- Write your name and roll number at the beginning of each program using comments.
- **Plagiarism:** Plagiarism is not allowed. If found plagiarized a case would be submitted against both parties to the DC committee.
- **Upload only cpp files, do not upload a zip file**

---

1 Employee Class

Write a class named Employee that has the following member variables:

- **name.** A string that holds the employee's name.
- **idNumber.** An int variable that holds the employee's ID number.
- **department.** A string that holds the name of the department where the employee works.
- **position.** A string that holds the employee's job title.

The class should have the following constructors:

- A constructor that accepts the following values as arguments and assigns them to the appropriate member variables: employee s name, employee s ID number, department, and position.
- A constructor that accepts the following values as arguments and assigns them to the appropriate member variables: employee's name and ID number. The department and position fields should be assigned an empty string (“”). A default constructor that assigns empty strings (“”) to the name, department, and position member variables, and 0 to the idNumber member variable.

Write appropriate mutator functions that store values in these member variables and accessor functions that return the values in these member variables.

Once you have written the class, write a separate program that creates an array of n Employee objects to hold the data, where n is taken as input from the user. The program should then display a menu to (1) allow the user to enter an Employee's record, (2) delete a record the array, and (3) find record(s) based on a criteria (one of these: name, idNumber, department, position).

Let your imagination decide how to better facilitate the user in finding all matching records.

2 Car Class

Write a class named *Car* that has the following member variables:

- **yearModel.** An *int* that holds the car's year model.
- **make.** A *string* that holds the make of the car.
- **speed.** An *int* that holds the car's current speed.

In addition, the class should have the following constructor and other member functions.

- **Constructor.** The constructor should accept the car's year model and make as arguments. These values should be assigned to the object's *yearModel* and *make* member variables. The constructor should also assign 0 to the *speed* member variable.
- **Accessor.** Appropriate accessor functions to get the values stored in an object's *yearModel*, *make* and *speed* member variables.
- **accelerate.** The *accelerate* function should add 5 to the *speed* member variable each time it is called.
- **brake.** The *brake* member function should subtract 5 from the *speed* member variable each time it is called.

Demonstrate the class in a program that creates a Car object with attributes taken as input from the user. The program should then display a menu to allow the user to drive the car. Obviously, there should be an option corresponding to the terminating the program. Also display the car speed every time the user accelerates or applies the brake.

3 Inventory Class

Design an *Inventory* class that can hold information and calculate data for items in a retail store's inventory. The class should have the following `private` member variables:

Variable Name	Description
itemNumber	An *int* that holds the item's item number.
quantity	An *int* for holding the quantity of the items on hand.
cost	A *double* for holding the wholesale per-unit cost of the item
totalCost	A *double* for holding the total inventory cost of the item (calculated as quantity times cost).

The class should have the following `public` member (interface) functions:

Member Function	**Description**
Default Constructor	Sets all the member variables to 0.
Constructor #2	Accepts an item's number, cost, and quantity as arguments. The function should copy these values to the appropriate member variables and then call the setTotalCost function.
setItemNumber	Accepts an integer argument that is copied to the itemNumber member variable.
setQuantity	Accepts an integer argument that is copied to the quantity member variable.
setCost	Accepts a double argument that is copied to the cost member variable.
setTotalCost	Calculates the total inventory cost for the item (quantity times cost) and stores the result in totalCost.
getItemNumber	Returns the value in itemNumber.
getQuantity	Returns the value in quantity.
getCost	Returns the value in cost.
getTotalCost	Returns the value in totalCost.

Write a menu driven program that uses this class to hold n items, where n is taken as input from the user. This program should allow the user to add an item, modify an item, find an item and delete an item. Let your imagination decide more details.