Programming Challenges

1. Date

Design a class called `Date`. The class should store a date in three integers: `month`, `day`, and `year`. There should be member functions to print the date in the following forms:

12/25/2018
December 25, 2018
25 December 2018

Demonstrate the class by writing a complete program implementing it.

Input Validation: Do not accept values for the day greater than 31 or less than 1. Do not accept values for the month greater than 12 or less than 1.

2. `Employee` Class

VideoNote
Solving the `Employee` Class Problem

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

- **`name`**—a string that holds the employee's name
- **`idNumber`**—a n `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 three `Employee` objects to hold the following data:

Name	ID Number	Department	Position
Susan Meyers	47899	Accounting	Vice President
Mark Jones	39119	IT	Programmer
Joy Rogers	81774	Manufacturing	Engineer

The program should store this data in the three objects and then display the data for each employee on the screen.

3. `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 variables.
- 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` 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, then calls the `accelerate` function five times. After each call to the `accelerate` function, get the current speed of the car and display it. Then, call the `brake` function five times. After each call to the `brake` function, get the current speed of the car and display it.

4. Patient Charges

Write a class named `Patient` that has member variables for the following data:

- First name, middle name, last name
- Address, city, state, and ZIP code
- Phone number
- Name and phone number of emergency contact

The `Patient` class should have a constructor that accepts an argument for each member variable. The `Patient` class should also have accessor and mutator functions for each member variable.

Next, write a class named `Procedure` that represents a medical procedure that has been performed on a patient. The `Procedure` class should have member variables for the following data:

- Name of the procedure
- Date of the procedure
- Name of the practitioner who performed the procedure
- Charges for the procedure

The `Procedure` class should have a constructor that accepts an argument for each member variable. The `Procedure` class should also have accessor and mutator functions for each member variable.

Next, write a program that creates an instance of the `Patient` class, initialized with sample data. Then, create three instances of the `Procedure` class, initialized with the following data:

Procedure #1:	Procedure #2:	Procedure #3:
Procedure name: Physical Exam	Procedure name: X-ray	Procedure name: Blood test
Date: Today's date	Date: Today's date	Date: Today's date
Practitioner: Dr. Irvine	Practitioner: Dr. Jamison	Practitioner: Dr. Smith
Charge: 250.00	Charge: 500.00	Charge: 200.00

The program should display the patient's information, information about all three of the procedures, and the total charges of the three procedures.

5. **RetailItem Class**

Write a class named RetailItem that holds data about an item in a retail store. The class should have the following member variables:

- **description**—a string that holds a brief description of the item
- **unitsOnHand**—an int that holds the number of units currently in inventory
- **price**—a double that holds the item's retail price

Write a constructor that accepts arguments for each member variable, 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 three RetailItem objects and stores the following data in them:

	Description	Units On Hand	Price
Item #1	Jacket	12	59.95
Item #2	Designer Jeans	40	34.95
Item #3	Shirt	20	24.95

6. **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 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.

Demonstrate the class in a driver program.

Input Validation: Do not accept negative values for item number, quantity, or cost.

7. TestScores Class

Design a TestScores class that has member variables to hold three test scores. The class should have a constructor, accessor, and mutator functions for the test score fields and a member function that returns the average of the test scores. Demonstrate the class by writing a separate program that creates an instance of the class. The program should ask the user to enter three test scores, which are stored in the TestScores object. Then the program should display the average of the scores, as reported by the TestScores object.

8. Circle Class

Write a Circle class that has the following member variables:

- **radius**—a double
- **pi**—a double initialized with the value 3.14159

The class should have the following member functions:

- **Default Constructor**—a default constructor that sets radius to 0.0
- **Constructor**—accepts the radius of the circle as an argument
- **setRadius**—a mutator function for the radius variable
- **getRadius**—an accessor function for the radius variable
- **getArea**—returns the area of the circle, which is calculated as

  ```
  area = pi * radius * radius
  ```
- **getDiameter**—returns the diameter of the circle, which is calculated as

  ```
  diameter = radius * 2
  ```
- **getCircumference**—returns the circumference of the circle, which is calculated as

  ```
  circumference = 2 * pi * radius
  ```

Write a program that demonstrates the Circle class by asking the user for the circle's radius, creating a Circle object, then reporting the circle's area, diameter, and circumference.

9. Population

In a population, the birth rate and death rate are calculated as follows:

Birth Rate = Number of Births ÷ Population
Death Rate = Number of Deaths ÷ Population

For example, in a population of 100,000 that has 8,000 births and 6,000 deaths per year, the birth rate and death rate are:

Birth Rate = 8,000 ÷ 100,000 = 0.08
Death Rate = 6,000 ÷ 100,000 = 0.06

Design a Population class that stores a population, number of births, and number of deaths for a period of time. Member functions should return the birth rate and death rate. Implement the class in a program.

Input Validation: Do not accept population figures less than 1, or birth or death numbers less than 0.