Assignment 1

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 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.

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 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, and 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.

3. 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

4. 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.
`setItemNumber`	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.

5. `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, and then reporting the circle's area, diameter, and circumference.

6. Coin Toss Simulator

Write a class named `Coin`. The `Coin` class should have the following member variable:

- A `string` named `sideUp`. The `sideUp` member variable will hold either “heads” or “tails” indicating the side of the coin that is facing up.

The `Coin` class should have the following member functions:

- A default constructor that randomly determines the side of the coin that is facing up (“heads” or “tails”) and initializes the `sideUp` member variable accordingly.
- A `void` member function named toss that simulates the tossing of the coin. When the toss member function is called, it randomly determines the side of the coin that is facing up (“heads” or “tails”) and sets the `sideUp` member variable accordingly.
- A member function named `getSideUp` that returns the value of the `sideUp` member variable.

Write a program that demonstrates the `Coin` class. The program should create an instance of the class and display the side that is initially facing up. Then, use a loop to toss the coin 20 times. Each time the coin is tossed, display the side that is facing up. The program should keep count of the number of times heads is facing up and the number of times tails is facing up, and display those values after the loop finishes.

7. Freezing and Boiling Points

The following table lists the freezing and boiling points of several substances.

Substance	**Freezing Point**	**Boiling Point**
Ethyl Alcohol	-173	172
Oxygen	-362	-306
Water	32	212

Design a class that stores a temperature in a `temperature` member variable and has the appropriate accessor and mutator functions. In addition to appropriate constructors, the class should have the following member functions:

- **`isEthylFreezing`**. This function should return the `bool` value `true` if the temperature stored in the `temperature` field is at or below the freezing point of ethyl alcohol. Otherwise, the function should return `false`.
- **`isEthylBoiling`**. This function should return the `bool` value `true` if the temperature stored in the `temperature` field is at or above the boiling point of ethyl alcohol. Otherwise, the function should return `false`.
- **`isOxygenFreezing`**. This function should return the `bool` value `true` if the temperature stored in the `temperature` field is at or below the freezing point of oxygen. Otherwise, the function should return `false`.
- **`isOxygenBoiling`**. This function should return the `bool` value `true` if the temperature stored in the `temperature` field is at or above the boiling point of oxygen. Otherwise, the function should return `false`.
- **`isWaterFreezing`**. This function should return the `bool` value `true` if the temperature stored in the `temperature` field is at or below the freezing point of water. Otherwise, the function should return `false`.

- **`isWaterBoiling`**. This function should return the `bool` value true if the temperature stored in the `temperature` field is at or above the boiling point of water. Otherwise, the function should return `false`.

Write a program that demonstrates the class. The program should ask the user to enter a temperature and then display a list of the substances that will freeze at that temperature and those that will boil at that temperature. For example, if the temperature is −20 the class should report that water will freeze and oxygen will boil at that temperature.

8. Trivia Game

In this programming challenge you will create a simple trivia game for two players. The program will work like this:

- Starting with player 1, each player gets a turn at answering five trivia questions. (There are a total of 10 questions.) When a question is displayed, four possible answers are also displayed. Only one of the answers is correct, and if the player selects the correct answer he or she earns a point.
- After answers have been selected for all of the questions, the program displays the number of points earned by each player and declares the player with the highest number of points the winner.

In this program you will design a `Question` class to hold the data for a trivia question. The `Question` class should have member variables for the following data:

- A trivia question
- Possible answer #1
- Possible answer #2
- Possible answer #3
- Possible answer #4
- The number of the correct answer (1, 2, 3, or 4)

The `Question` class should have appropriate constructor(s), accessor, and mutator functions.

The program should create an array of 10 `Question` objects, one for each trivia question. Make up your own trivia questions on the subject or subjects of your choice for the objects.