1. Write a class named **`Painting`** that has the following member variables:

   i. **`artist`**: A string that holds the painter's name who draws the painting.

   ii. **`yearPainted`**: An **`int`** that holds the year the painting was drawn.

   iii. **`auctionPrice`**: A **`double`** that holds the painting's auction price.

   In addition, the class have the following member functions:

   i. A constructor that accepts the artist and the year the painting was drawn as arguments. These values should be assigned to the object's **`artist`** and **`yearPainted`** member variables. The constructor should also assign **`1000`** to the **`auctionPrice`** member variables.

   ii. Appropriate mutator functions to store values to an object's **`artist`**, **`yearPainted`** and **`auctionPrice`** member variables.

   iii. Appropriate accessor functions to get the values stored in an object's **`artist`**, **`yearPainted`** and **`auctionPrice`** member variables.

   iv. The **`auction`** function should add **`500`** to the **`auctionPrice`** member variable each time it is called.

   v. The **`displayPrice`** function should display the **`auctionPrice`** value each time it is called.

   Write the **`main()`** function that does the following tasks:

   i. Creates a **`Painting`** object then,

   ii. Calls the **`auction`** function five times.

   iii. After each call to the **`auction`** function, get the current **`auctionPrice`** of the painting and display it.

2. Write a class named **`Stock`** that contains the following:

   i. A **`string`** data field named **`code`** for the stock's code.

   ii. A **`string`** data field named **`sName`** for the stock's name.

   iii. A **`double`** data field named **`closePrice`** that stores the stock price for the previous trading day.

   iv. A **`double`** data field named **`currentPrice`** that stores the current stock price.

   v. A constructor that creates a stock with specified code, name and **`closePrice`**.

vi. The accessor functions for all data fields.

vii. The mutator functions for **`closePrice`** and **`currentPrice`**.

viii. A function named **`changePercent()`** that returns the percentage changed from **`closePrice`** to **`currentPrice`**.

Write a **`main()`** function that do the following tasks:

i. Create a **`Stock`** object with the stock code "**`MBB`**", the name "**`Maybank`**", and the previous closing price of 100.

ii. Set a new current price to 90.

iii. Display the price change percentage.

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

i. **`name`**: A **`string`** that holds the employee's name.

ii. **`idNumber`**: An int variable that holds the employee's ID number.

iii. **`department`**: a string that holds the name of the department where the employee works.

iv. **`position`**: a string that holes the employee's job title.

v. 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.

vi. 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 (" ").

vii. A default constructor that assigns empty strings (" ") to the name, department, and position member variables, and 0 to the idNumber member variable.

viii. Write appropriate mutator functions that store values in these member variables.

ix. Write appropriate accessor functions that return the values in these member variables.

Write a **`main()`** function that do the following tasks:

i. Create an object called **`emp1`** that have the following data:
Name: Syed Mahadi
ID Number: 47899
Department: Accounting
Position: Vice President

ii. Display the data for each employee on the screen.

4. Write a complete C++ program that contains a class called **`Breads`**. This class is used for holding data about bread products sold by a bakery. The class should have the following *private* member variables (attributes):

Attribute	Description
`code`	is of type **`int`** that holds the product's code number.
`price`	is of type **`double`** that holds the price of the product.
`name`	is of type **`string`** that holds the name of the bread based on the values in Table 5.1.

The class should also have the following *public* member functions (methods):

Method	Description
Default Constructor	sets all the attributes to **`0`**.
Overloaded Constructor	sets all the attributes from arguments.
Destructor	does nothing.
`setCode`	is a mutator function that sets the **`code`** attribute from an **`int`** argument.
`setPrice`	is a mutator function that sets the **`price`** attribute from a **`double`** argument.
`setBreadname`	is a mutator function that sets the **`name`** attribute (a string value) from the **`code`**.
`getCode`	is an accessor function that returns the value of **`code`** attribute.
`getPrice`	is an accessor function that returns the value of **`price`** attribute.
`getBreadname`	is an accessor function that returns the value of **`name`** attribute.

Table 5.1 Bread Names according to Bread Codes

Bread Code	Bread Name
101	Plain Croissant
102	Cream-filled Croissant
103	Sausage Croissant
202	Plain Bun
204	Sambal Bun
205	Cheese Bun
301	Plain Braid
305	Cheese Braid
306	Sugar Braid

In the main function of the program:

i. declare an array to hold a list of up to 100 objects of class **`Breads`**.

ii. prompt the user to enter the number of bread products to be recorded.

iii. set the attributes of each object in the array using the mutator functions. Data should be read from the keyboard first. Your program is expected to use a loop for doing this.

iv. add another **`Breads`** object to the array. The objects's **`code`** and **`price`** attributes are set to 100 and 1.00 respectively. Your program is expected to use the overloaded constructor for creating this object.

v. Finally, print the bread information in tabular form as shown in the following figure. Your program should use another loop for doing this part. Your program should also calculate and print the total price of the breads as in Figure 5.5.

The **text in bold fonts** in Figure 5.5 below represents user inputs from the keyboard.

```
How many bread types to enter? 4

Enter the information of bread #1
Bread Code: 102
Bread Name: Cream-filled Croissant
Price(RM): 3.50

Enter the information of bread #2
Product Code: 204
Bread Name: Sambal Bun
Price(RM): 2.00

Enter the information of bread #3
Product Code: 305
Bread Name: Cheese Braid
Price(RM): 3.00
```

```
Enter the information of bread #4
Product Code: 306
Bread Name: Sugar Braid
Price(RM): 2.50

No  Bread Code          Bread Name             Price(RM)

1       102         Cream-filled Croissant       3.50
2       204         Sambal Bun                   2.00
3       305         Cheese Braid                 3.00
4       306         Sugar Braid                  2.50

Average Price(RM):  11.00
```

Figure 5.5