Question Paper 5

1. Code Completion

(a). Define the class `Book` for the given code.

```
int main()
{
    Book b1;
    b1.readBook();
    b1.displayBook();
    return 0;
}
```

Ans.: Here the a definition of the class `Book` according to the code given above:

```
#include <iostream>
using namespace std;

class Book
{
    string title, author;

public:
    Book()
    {
        title = "Unknown";
        author = "Unknown";
    }

    void readBook()
    {
        cout << "Reading " << title << " by " << author <<
endl;
```

```
    }

    void displayBook()
    {
        cout << "Title : " << title << endl;
        cout << "Author: " << author << endl;
    }
};

int main()
{
    Book b1;
    b1.readBook();
    b1.displayBook();
    return 0;
}
```

(b). How can you access the private attribute `balance` of a class `Account` outside the class?

```
class Account
{
private:
    float balance;
};
```

Ans.: We can access the private attribute `balance` from anywhere using a getter function like below:

```
#include <iostream>
using namespace std;

class Account
{
private:
```

```
    float balance;

public:
    float getBalance()
    {
        return balance;
    }
};

int main()
{
    Account account;
    cout << account.getBalance() << endl;
    return 0;
}
```

(c). Define the functions of the `Distance` class outside the class.

```
class Distance {
public:
    int feet;
    float inches;
    void addData(int f, float in);
    void showData();
};
```

Ans.: Here is how we can define the functions of the given class outside it:

```
#include <iostream>
using namespace std;

class Distance
{
public:
```

```
    int feet;
    float inches;
    void addData(int f, float in);
    void showData();
};

void Distance::addData(int f, float in)
{
    feet = f;
    inches = in;
}

void Distance::showData()
{
    cout << "Feet  : " << feet << endl;
    cout << "Inches: " << inches << endl;
}

int main()
{
    Distance d;
    d.addData(10, 5.7);
    d.showData();
    return 0;
}
```

2. Problem Solving

(a). Design a `Product` class

- The class should have attributes for its name, price, and quantity. Implement a function called `setProduct()` within the class to input the product details from the user. Additionally, create a function named `printProduct()` to display the product details.

- Ensure that `setProduct()` is not directly called in the `main()` function. Add a destructor to print **"Product object destroyed"** when an object is destroyed.

Ans.: Here is a C++ program to design a `Product` class with the instructions given above:

```cpp
#include <iostream>
using namespace std;

class Product
{
    string name;
    float price;
    int quantity;

public:
    Product(string n, float p, int q)
    {
        setProduct(n, p, q);
    }

    ~Product()
    {
        cout << "Product object destroyed" << endl;
    }
```

```
    void setProduct(string n, float p, int q)
    {
        name = n;
        price = p;
        quantity = q;
    }

    void printProduct()
    {
        cout << "Name:     " << name << endl;
        cout << "Price:    " << price << endl;
        cout << "Quantity: " << quantity << endl;
    }
};

int main()
{
    Product p("Biscuits", 10.00, 1);
    p.printProduct();
    return 0;
}
```

Output: The above C++ code yields the following output in the terminal:

```
Name:     Biscuits
Price:    10
Quantity: 1
Product object destroyed
```

(b). Implement an `Account` class

- Implement a class called `Account` with attributes like **account holder name**, **account number**, **account type**, and **balance**. Use a parameterized constructor to set these attributes when a new account is created.

- If attributes are not provided, use the default constructor to set the values:

```
account holder name = "Not Assigned"
account number = 0
account type = "Savings"
balance = 0.0
```

Ans.: Here is a C++ demonstrating an implementation of an `Account` class that satisfies the given conditions:

```cpp
#include <iostream>
using namespace std;

class Account
{
    string accHolderName;
    int accNumber;
    string accType;
    double balance;

public:
    Account()
    {
        accHolderName = "Not Assigned";
        accNumber = 0;
        accType = "Savings";
        balance = 0.0;
    }

    Account(string hn, int an, string at, double b)
    {
        accHolderName = hn;
        accNumber = an;
        accType = at;
        balance = b;
    }
```

```
    void display()
    {
        cout << endl;
        cout << "Account Information:" << endl;
        cout << "Holder Name: " << accHolderName << endl;
        cout << "A/C Number : " << accNumber << endl;
        cout << "A/C Type   : " << accType << endl;
        cout << "Balance    : " << balance << "৳" << endl;
    }
};

int main()
{
    Account a1;
    Account a2("Shahriar", 408, "Savings", 42900.50);

    a1.display();
    a2.display();
    return 0;
}
```

Output: The above C++ code yields the following output in the terminal:

```
Account Information:
Holder Name: Not Assigned
A/C Number : 0
A/C Type   : Savings
Balance    : 0৳

Account Information:
Holder Name: Shahriar
A/C Number : 408
A/C Type   : Savings
Balance    : 42900.5৳
```

Code

You can find all the code snippets **here**.