Question Paper 6

1. Short Questions

- Write the general definition of a `class`.
- Differentiate between **private** and **public** members of a `class` with a suitable example.

Ans.: Classes and objects are the basic building block that leads to Object-Oriented programming in C++. A `class` is a **user-defined data type**, which holds its own **data members** and **member functions**, which can be accessed and used by creating an instance of that class. A C++ class is like a blueprint for an object.

In C++, we have **private** and **public** keywords that modifies the accessibility of **classes**, **methods**, and **other members**.

All the class members declared under the **public** access modifier will be available to everyone. For example:

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

class Person
{
public:
    string name;
    int age;

    Person(string n, int a)
    {
        name = n;
        age = a;
    }
};
```

```
int main()
{
    Person p1("Shahriar", 21);
    cout << p1.name << endl;
    cout << p1.age << endl;
    return 0;
}
```

Here we can access the data members (**name**, **age**) of a `Person` class outside it. We can now modify the above example to demonstrate the effect of the **private** keyword:

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

class Person
{
private:
    string name;
    int age;

public:
    Person(string n, int a)
    {
        name = n;
        age = a;
    }

    string getName()
    {
        return name;
    }

    int getAge()
    {
```

```
        return age;
    }
};

int main()
{
    Person p1("Shahriar", 21);
    cout << p1.getName() << endl;
    cout << p1.getAge() << endl;
    return 0;
}
```

The class members declared **private** can be accessed only by the member functions inside the class. Only the member functions or the **friend functions/classes** are allowed to access the private data members of the class.

This is why we implemented two getter methods to retrieve the **name** and **age** from the instance of the `Person` class.

2. Short Question

Describe different types of `constructors` with examples in terms of Object Oriented Programming.

Ans.: `Constructor` in Object Oriented Programming is a special method that is invoked automatically **at the time an object of a class is created**. It is used to initialize the data members of new objects generally.

Types of Constructor:

- **Default Constructor:** A default constructor is a constructor that doesn't take any argument. It has no parameters. It is also called a zero-argument constructor.

```
class Person
{
    string name;

public:
    Person()
    {
        name = "Unknown";
    }
};
```

- **Parameterized Constructor:** Parameterized constructors make it possible to pass arguments to constructors. Typically, these arguments help initialize an object when it is created.

```
class Person
{
    string name;
    int age;
```

```
public:
    Person(string n, int a)
    {
        name = n;
        age = a;
    }
};
```

- **Copy Constructor:** A copy constructor is a **member function** that initializes an object using another object of the same class. Copy constructor takes a reference to an object of the same class as an argument.

```
class Person
{
    string name;

public:
    Person(const Person &p)
    {
        name = p.name;
        cout << "Copy constructor called!" << endl;
    }
};
```

3. A `Circle` class

- Design a class named `Circle`. Construct **three circle objects** with radius **2.0**, **12**, and **24** and display the **radius** and **area** of each object.
- Copy the `radius` of the 2nd object into a new object using a user defined **copy constructor**. A `getArea()` function is used to return the `area` of a circle.

Now implement the code using C++ mechanism.

Ans.: Here is C++ program that implements a class `Circle` and satisfies the given conditions:

```
#include <iostream>
#define PI 3.14159
using namespace std;

class Circle
{
    float radius;

public:
    Circle(float r)
    {
        radius = r;
    }

    Circle(const Circle &c)
    {
        radius = c.radius;
        cout << "Copy constructor called!" << endl;
    }

    float getArea()
    {
```

```
        return PI * (radius * radius);
    }

    void display()
    {
        cout << "Radius: " << radius << endl;
        cout << "Area  : " << getArea() << endl;
        cout << endl;
    }
};

int main()
{
    Circle c1(2.0);
    c1.display();

    Circle c2(12);
    c2.display();

    Circle c3(24);
    c3.display();

    Circle c4 = c2;
    c4.display();
    return 0;
}
```

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

```
Radius: 2
Area  : 12.5664

Radius: 12
Area  : 452.389

Radius: 24
Area  : 1809.56
```

```
Copy constructor called!
Radius: 12
Area  : 452.389
```

4. The `Intake99` class

- A class named `Intake99` has **three private members** named `name`, `cgpa`, and `id`. The `name` and `id` are of **string** type and **cgpa** is a **double** data type.
- A member function `set_val()` is used to set the values and another member function `show_val()` is used to show the values of the student.

Now create **15 students** and display all the information for the **15 students** using an array of objects.

Ans.:

```
#include <iostream>
#define maxStudents 15
using namespace std;

class Intake99
{
    string name;
    double cgpa, id;

public:
    Intake99(string n, double c, double i)
    {
        name = n;
        cgpa = c;
        id = i;
    }

    void set_val(string n, double c, double i)
    {
        name = n;
        cgpa = c;
        id = i;
```

```
    }

    void show_val()
    {
        cout << "Name: " << name;
        cout << ", ID: " << id;
        cout << ", CGPA: " << cgpa << endl;
    }
};

int main()
{
    Intake99 students[maxStudents] = {
        Intake99("Shahriar", 3.55, 408),
        Intake99("Redowan", 3.58, 349),
        Intake99("Manzirul", 3.77, 357),
        Intake99("Hakim", 4.00, 360),
        Intake99("Arefin", 3.98, 347),
        Intake99("Fikrat", 3.87, 328),
        Intake99("Rafi", 3.87, 346),
        Intake99("Saif", 3.67, 361),
        Intake99("Shawon", 3.68, 356),
        Intake99("John", 3.81, 331),
        Intake99("Mimiya", 3.95, 339),
        Intake99("Dristy", 3.98, 343),
        Intake99("Behesti", 3.77, 338),
        Intake99("Protiva", 3.68, 348),
        Intake99("Jannatul", 3.48, 108)};

    for (int i = 0; i < maxStudents; i++)
        students[i].show_val();

    return 0;
}
```

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

```
Name: Shahriar, ID: 408, CGPA: 3.55
Name: Redowan, ID: 349, CGPA: 3.58
Name: Manzirul, ID: 357, CGPA: 3.77
Name: Hakim, ID: 360, CGPA: 4
Name: Arefin, ID: 347, CGPA: 3.98
Name: Fikrat, ID: 328, CGPA: 3.87
Name: Rafi, ID: 346, CGPA: 3.87
Name: Saif, ID: 361, CGPA: 3.67
Name: Shawon, ID: 356, CGPA: 3.68
Name: John, ID: 331, CGPA: 3.81
Name: Mimiya, ID: 339, CGPA: 3.95
Name: Dristy, ID: 343, CGPA: 3.98
Name: Behesti, ID: 338, CGPA: 3.77
Name: Protiva, ID: 348, CGPA: 3.68
Name: Jannatul, ID: 108, CGPA: 3.48
```

5. A Tale of Two Boxes

- Define a class named `Box1` which has **two private data members**, one is area as **double** type another is color type **string**. Also define another class named `Box2` which has the same **private data members** as `Box1` .

- A **parameterized constructor** is used to set the data members of the two classes. Now define a **non-member function** named `CompareBox()` which takes **two objects** as parameters and compares the area of the `Box1` and `Box2` .

Ans.: Here is a C++ program that satisfies the above conditions:

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

class Box1
{
    double area;
    string color;

public:
    Box1(double a, string c)
    {
        area = a;
        color = c;
    }

    double getArea()
    {
        return area;
    }
};

class Box2
{
    double area;
```

```
    string color;

public:
    Box2(double a, string c)
    {
        area = a;
        color = c;
    }

    double getArea()
    {
        return area;
    }
};

void CompareBox(Box1 a, Box2 b)
{
    if (a.getArea() > b.getArea())
        cout << "Box1 has a larger area than Box2.";
    else
        cout << "Box2 has a larger area than Box1.";
}

int main()
{
    Box1 b1(1248, "red");
    Box2 b2(512, "blue");
    CompareBox(b1, b2);
    return 0;
}
```

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

```
Box1 has a larger area than Box2.
```

6. Dineout at KFC

- Suppose there is a restaurant named KFC. You have to choose your food item. You are asked to write a class `KFC` where the function `chooseFood()` takes the name of foods as the argument.

- The number of food items may vary from **three (3)** to **five (5)** and from person to person. Complete the task using function overloading concept.

Ans.: Here is a C++ program that satisfies the above conditions:

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

class KFC
{
    string orders[5];

public:
    void chooseFood(string f1, string f2, string f3)
    {
        orders[0] = f1;
        orders[1] = f2;
        orders[2] = f3;
    }

    void
chooseFood(string f1, string f2, string f3, string f4)
    {
        orders[0] = f1;
        orders[1] = f2;
        orders[2] = f3;
        orders[3] = f4;
    }

    void
chooseFood(string f1, string f2, string f3, string f4,
```

```
string f5)
    {
        orders[0] = f1;
        orders[1] = f2;
        orders[2] = f3;
        orders[3] = f4;
        orders[4] = f5;
    }

    void displayOrders()
    {
        for (int i = 0; i < 5; i++)
            if (orders[i] != "")
                cout << orders[i] << endl;
    }
};

int main()
{
    KFC restaurant;

    restaurant.chooseFood("Pizza", "Burger", "Coke");
    restaurant.displayOrders();
    cout << endl;

    restaurant.chooseFood("Meatballs", "Burger", "Mirinda",
"Desert");
    restaurant.displayOrders();
    cout << endl;

    restaurant.chooseFood("Sweetpuffs", "Chocobursts",
"Red Velvet", "Desert", "Takeout");
    restaurant.displayOrders();

    return 0;
}
```

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

```
Pizza
Burger
Coke

Meatballs
Burger
Mirinda
Desert

Sweetpuffs
Chocobursts
Red Velvet
Desert
Takeout
```

Code

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