Question Paper 1

1. Write down the answers.

(a). What is Access Modifier in Object Oriented Programming?

Ans.: Access modifiers (or access specifiers) are **keywords** in object-oriented programming languages that set the accessibility of **classes**, **methods**, and **other members**. Access modifiers are a specific part of programming language syntax used to facilitate the **encapsulation** of components.

In C++, there are **three** access modifiers:

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

2. **Private**: The class members declared as **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.

3. **Protected:** The **protected** access modifier is similar to the private access modifier in the sense that it can't be accessed outside of its class unless with the help of a **friend class**. The difference is that the class members declared as **protected** can be accessed by any subclass (derived class) of that class as well.

(b). Write down the differences between Constructor and Destructor.

Ans.: Here are the differences between Constructor and Destructor:

Constructor	Destructor
Constructor is a member function that has the same name as the name of the class.	Destructors are typically used to deallocate memory.
When the object is created, a constructor is called automatically.	When the program gets terminated, the destructor is called automatically.
A constructor allows an object to initialize some of its value before it is used.	A destructor allows an object to execute some code at the time of its destruction.
There can be various constructors in a class	There is constantly a single destructor in the class
Can be overloaded.	Cannot be overloaded.
Receives arguments.	Does not receive any argument.

2. Explain Union in C++ with a suitable example.

Ans.: In C++, a union is a **user-defined data type** in which we can define members of different data types just like structures. The the member variables in a union share the same memory location. The size of the union is equal to the size of the largest data type.

A union can be used to achieve memory efficiency when the available memory is limited. It is used to encapsulate different types of data members.

Here is a suitable example that demonstrates the use of a union:

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

union student
{
    int sid;
    float cgpa;
};

int main()
{
    union student std1;

    std1.sid = 408;
    cout << &std1.sid << endl;

    std1.cgpa = 4.86;
    cout << &std1.cgpa << endl;
    return 0;
}
```

We know that the members of the union share the same memory location. So, if we print their memory addresses, they will be the same.

```
    cout << &std1.sid << endl; // 0x5219ffbdc
    cout << &std1.cgpa << endl; // 0x5219ffbdc
```

3. Design a C++ program.

Create a C++ program that defines a class Student with the following private data members: name, roll, and marks in three subjects. The class should have the following public member functions:

- **setDetails()**: This method takes tthe student's name, roll, and marks for three subjects as parameters, and sets them to the respective data members.
- **getTotalMarks()**: This method returns the total marks obtained by the student.
- **getAverageMarks()**: This method returns the average of the student's marks.
- **display()**: This method displays the student's name, roll, total marks and average marks.

In the `main()` function, create an object of the `Student` class, set the student's details, and display their details including the total and average marks.

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

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

class Student
{
private:
    string name;
    int roll;
    float marks[3];

public:
    void setDetails(string n, int r, float m[3])
    {
```

```cpp
        name = n;
        roll = r;
        marks[0] = m[0];
        marks[1] = m[1];
        marks[2] = m[2];
    }

    float getTotalMarks()
    {
        return marks[0] + marks[1] + marks[2];
    }

    float getAverageMarks()
    {
        return getTotalMarks() / 3;
    }

    void display()
    {
        cout << "Name         : " << name << endl;
        cout << "Roll No.     : " << roll << endl;
        cout << "Total Marks  : " << getTotalMarks();
        cout << endl;
        cout << "Average Marks: " << getAverageMarks();
        cout << endl;
    }
};

int main()
{
    Student std1;
    float marks[3] = {96.25, 56.00, 77.75};
    std1.setDetails("Shahriar", 408, marks);
    std1.display();
}
```

Output: The code yields the following output in the terminal:

```
Name         : Shahriar
Roll No.     : 408
Total Marks  : 230
Average Marks: 76.6667
```

References

- **Wikipedia:** Access modifiers
- **GeeksforGeeks:** Access Modifiers in C++
- **Byjus:** Difference Between Constructor and Destructor in C++
- **GeeksforGeeks:** C++ Unions