Class Test 1

1. Theory-based Question

What is **Polymorphism** in C++? Can a function be marked as **inline** if it is overloaded? Explain your answer with an example.

Ans.: The word "polymorphism" means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form. In C++, function overloading is a great example of polymorphism.

Function overloading is a feature of object-oriented programming where two or more functions can have the same name but different parameters. For example-

```
int add(int a, int b) {
    return a + b;
}

int add(int a, int b, int c) {
    return a + b + c;
}

double add(double a, double b, double c) {
    return a + b + c;
}
```

We know that we can use the `inline` keyword to mark a function inline,

```
inline int add(int a, int b){
    return a + b;
}
```

Inlining a function is ultimately decided by the compiler as a request, not a command. We can use the `inline` keyword to explicitly mark a function as inline, even if it's overloaded.

2. Problem Solving

Write a C++ program to create a class `Student` that stores a student's name, marks in three subjects, and includes the following:

- A **default constructor** that sets the `name` to "Unknown" and all `marks` to zero.
- A **parameterized constructor** to set custom values for the student's `name` and `marks` .
- Getters and setters for the `name` and `marks` .
- A method `calculateTotalMarks()` to calculate and return the total marks (sum of the three subjects).

Create two objects of the `class` , one using the **default constructor** and one using the **parameterized constructor**. Call the `calculateTotalMarks()` method for both objects and display the total marks.

Ans.: Here is the C++ program of a class named `Student` that satisfies the conditions above:

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

class Student
{
    string name;
    int marks1, marks2, marks3;

public:
    Student()
    {
        name = "Unknown";
        marks1 = 0;
```

```
        marks2 = 0;
        marks3 = 0;
    }

    Student(string n, int m1, int m2, int m3)
    {
        name = n;
        marks1 = m1;
        marks2 = m2;
        marks3 = m3;
    }

    string getName()
    {
        return name;
    }

    void getMarks()
    {
        cout << marks1 << endl;
        cout << marks2 << endl;
        cout << marks3 << endl;
    }

    void setName(string n)
    {
        name = n;
    }

    void setMarks(int m1, int m2, int m3)
    {
        marks1 = m1;
        marks2 = m2;
        marks3 = m3;
    }

    int calculateTotalMarks()
    {
```

```
        return marks1 + marks2 + marks3;
    }
};

int main()
{
    Student s1;
    Student s2("Shahriar", 60, 97, 88);

    cout << "Name:        " << s1.getName() << endl;
    cout << "Total Marks: " << s1.calculateTotalMarks() <<
endl;
    cout << endl;
    cout << "Name:        " << s2.getName() << endl;
    cout << "Total Marks: " << s2.calculateTotalMarks() <<
endl;
    return 0;
}
```

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

```
Name:        Unknown
Total Marks: 0

Name:        Shahriar
Total Marks: 245
```

3. Output Prediction

Consider the following C++ code and predict the output:

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

class Vehicle
{
    string type;
    int speed;

public:
    Vehicle(string t, int s)
    {
        type = t;
        speed = s;
        cout << type << " with speed " << speed << " km/h
is being created." << endl;
    }

    void accelerate(int amount)
    {
        speed += amount;
    }

    void showSpeed()
    {
        cout << type << " current speed: " << speed << "
km/h." << endl;
    }

    ~Vehicle()
    {
        cout << "Destructor: " << type << " with speed " <<
speed << " km/h is being destroyed." << endl;
    }
```

```
};

int main()
{
    Vehicle v1("Car", 100);
    v1.accelerate(20);
    Vehicle v2("Bike", 50);
    v2.accelerate(10);
    v2.showSpeed();
    v2 = v1;
    v2.showSpeed();
    return 0;
}
```

Ans.: The code yields the following output in the terminal:

```
Car with speed 100 km/h is being created.
Bike with speed 50 km/h is being created.
Bike current speed: 60 km/h.
Car current speed: 120 km/h.
Destructor: Car with speed 120 km/h is being destroyed.
Destructor: Car with speed 120 km/h is being destroyed.
```

References

- **GeeksforGeeks**: Polymorphism in C++
- **GeeksforGeeks**: Inline Functions in C++