Question Paper 4

1. What is Inlining in C++?

Give an example of automatic inlining.

Ans.: An **inline function** is a function that is expanded in line when it is called. When the inline function is called whole code of the inline function gets inserted or substituted at the point of the inline function call.

We know that all the functions defined inside a `class` are **implicitly** inline. So, here is an example code of automatic inlining using a `class` :

```
class Samp
{
public:
    int getData(int id)
    {
        // this function is automatically inline
        // function body
    }
};
```

2. Problem Solving

Write a C++ program to create a class `Rectangle` that stores the **length** and **width** of a rectangle. The class should include:

- A default constructor that sets both `length` and `width` to 1.
- A parameterized constructor to set custom values for `length` and `width`.
- Getters and setters for both `length` and `width`.
- A method named `calculateArea()` that calculates and returns the area of the rectangle (`length × width`).

Create two objects of the class, one using the **default constructor** and the other using the **parameterized constructor**. Call the `calculateArea()` method for both objects and display the area.

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

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

class Rectangle
{
    int length;
    int width;

public:
    Rectangle()
    {
        length = 1;
        width = 1;
```

```
    }

    Rectangle(int l, int w)
    {
        length = l;
        width = w;
    }

    void setLength(int l)
    {
        length = l;
    }

    void setWidth(int w)
    {
        width = w;
    }

    int getLength()
    {
        return length;
    }

    int getWidth()
    {
        return width;
    }

    int calculateArea()
    {
        return length * width;
    }
};

int main()
{
    Rectangle r1;
    Rectangle r2(64, 32);
```

```
    cout << "Area of r1: " << r1.calculateArea() << endl;
    cout << "Area of r2: " << r2.calculateArea() << endl;
    return 0;
}
```

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

```
Area of r1: 1
Area of r2: 2048
```

3. Output Prediction

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

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

class Car
{
public:
    Car()
    {
        cout << "Default Constructor: A car is being
built." << endl;
    }

    Car(string model)
    {
        cout << "Parameterized Constructor: Car model " <<
model << " is being built." << endl;
    }

    ~Car()
    {
        cout << "Destructor: The car is being destroyed."
        << endl;
    }

    void start()
    {
        cout << "The car engine is starting." << endl;
    }
};

int main()
{
    Car c1;             // Line 1
```

```
    Car c2("Sedan");    // Line 2
    c1.start();         // Line 3
    c2.start();         // Line 4
    return 0;
}
```

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

```
Default Constructor: A car is being built.
Parameterized Constructor: Car model Sedan is being built.
The car engine is starting.
The car engine is starting.
Destructor: The car is being destroyed.
Destructor: The car is being destroyed.
```

References

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