Class Test 1

1. Short Questions

- What is inlining in C++?
- Give an example of automatic inlining.

Ans.: Inlining is a manual or compiler optimization that replaces a function call site with the body of the called function.

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 automatically inlined. 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 `Employee` that stores an employee's name and their hourly wage. The class should include:

- A **default constructor** that sets both the `name` to "Unknown" and the hourly wage to zero.
- A **parameterized constructor** to set custom values for the student's `name` and `wage`.
- Getters and setters for the `name` and `wage`.
- A method `calculateWeeklyPay(int hoursWorked)` that calculates the employee's weekly pay and returns the amount.

Create two objects of the `class`, one using the **default constructor** and one using the **parameterized constructor**. Call the `calculateWeeklyPay()` method for both objects and display the weekly pay, assuming both work 40 hours a week.

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

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

class Employee
{
    string name;
    int wage;

public:
    Employee()
    {
        name = "Unknown";
```

```
        wage = 0;
    }

    Employee(string n, int w)
    {
        name = n;
        wage = w;
    }

    string getName()
    {
        return name;
    }

    int getHourlyWage()
    {
        return wage;
    }

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

    void setHourlyWage(int w)
    {
        wage = w;
    }

    int calculateWeeklyPay(int hoursWorked)
    {
        return wage * hoursWorked;
    }
};

int main()
{
    Employee e1;
```

```
    Employee e2("Shahriar", 35);

    cout << "Employee Name: " << e1.getName() << endl;
    cout << "Weekly Pay   : " << e1.calculateWeeklyPay(40)
<< "৳" << endl;
    cout << endl;
    cout << "Employee Name: " << e2.getName() << endl;
    cout << "Weekly Pay   : " << e2.calculateWeeklyPay(40)
<< "৳" << endl;
    return 0;
}
```

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

```
Employee Name: Unknown
Weekly Pay   : 0৳

Employee Name: Shahriar
Weekly Pay   : 1400৳
```

Output Prediction

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

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

class Smartphone
{
public:
    Smartphone()
    {
        cout << "Default Constructor: A new smartphone is
being initialized." << endl;
    }

    Smartphone(string brand, double price)
    {
        cout << "Parameterized Constructor: Smartphone
brand " << brand << " priced at $" << price << " is being
initialized." << endl;
    }

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

    void call(string number)
    {
        cout << "Calling " << number << " from the
smartphone." << endl;
    }
};

int main()
```

```
{
    Smartphone s1;                    // Line 1
    Smartphone s2("iPhone", 999.99); // Line 2
    s2.call("123-456-7890");          // Line 3
    s1.call("987-654-3210");          // Line 4
    return 0;
}
```

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

```
Default Constructor: A new smartphone is being initialized.
Parameterized Constructor: Smartphone brand iPhone priced
at $999.99 is being initialized.
Calling 123-456-7890 from the smartphone.
Calling 987-654-3210 from the smartphone.
Destructor: The smartphone is being destroyed.
Destructor: The smartphone is being destroyed.
```

References

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