Subject:- C++
LAB Assignment - 7

1.Write a C++ program to demonstrate operator overloading using friend functions. The program should perform the following tasks:
Class Definition:
Define a class Rectangle with the following private data members:
length (integer)
width (integer)
Include a default constructor that initializes both length and width to 0.
Operator Overloading:
Overload the >> operator using a friend function to take input for the length and width of a rectangle.
Overload the << operator using a friend function to output the length and width of a rectangle.
Main Function:
In the main() function, create an object of the Rectangle class.
Use the overloaded >> operator to input the dimensions of the rectangle.
Use the overloaded << operator to display the dimensions of The rectangle.

```
#include <iostream>

class Rectangle {
private:
    int length;  // Length of the rectangle
    int width;   // Width of the rectangle

public:
    // Default constructor that initializes length and width to 0
    Rectangle() : length(0), width(0) {}

    // Friend function to overload the >> operator for input
    friend std::istream& operator>>(std::istream& in, Rectangle& rect) {
        std::cout << "Enter length: ";
```

```
        in >> rect.length;
        std::cout << "Enter width: ";
        in >> rect.width;
        return in;
    }

    // Friend function to overload the << operator for output
    friend std::ostream& operator<<(std::ostream& out, const Rectangle& rect) {
        out << "Rectangle dimensions:\n";
        out << "Length: " << rect.length << "\n";
        out << "Width: " << rect.width << "\n";
        return out;
    }
};

int main() {
    Rectangle rect;  // Create a Rectangle object

    // Use the overloaded >> operator to input the dimensions
    std::cin >> rect;

    // Use the overloaded << operator to display the dimensions
    std::cout << rect;

    return 0;
}
```

Output:

```
Enter length: 5
Enter width: 10
Rectangle dimensions:
Length: 5
Width: 10
```

2.Write a C++ program to illustrate type conversion from a basic data

type (int) to a class type using a parameterized constructor. The program should perform the following tasks:

Class Definition:

Define a class Time with the following private data members:

- **hour (integer)**
- **mins (integer)**

Include a default constructor that initializes hour and mins to 0.

Include a parameterized constructor that takes an integer value representing time in minutes. This constructor should convert the given minutes into hours and minutes, storing the result in the hour and mins data members.

Display Function:

Implement a Display() function in the Time class that prints the time in the format Time = X hrs and Y mins, where X is the value of hour and Y is the value of mins.

Main Function:

In the main() function, create an object of the Time class.

Initialize an integer variable dur with a value (e.g., 95 minutes).

Convert the integer dur into a Time object using the parameterized constructor.

```
#include <iostream>

class Time {
private:
    int hour;  // Stores hours
    int mins;  // Stores minutes

public:
    // Default constructor that initializes hour and mins to 0
    Time() : hour(0), mins(0) {}

    // Parameterized constructor that takes time in minutes
    Time(int totalMinutes) {
        hour = totalMinutes / 60;  // Convert total minutes to hours
        mins = totalMinutes % 60;   // Get the remaining minutes
    }
```

```
    // Function to display the time in the format "Time = X hrs and Y mins"
    void Display() const {
        std::cout << "Time = " << hour << " hrs and " << mins << " mins" <<
std::endl;
    }
};

int main() {
    int dur = 95;  // Initialize an integer variable with value in minutes

    // Create a Time object using the parameterized constructor
    Time timeObj(dur);

    // Display the converted time
    timeObj.Display();

    return 0;
}
```

Output:

```
Time = 1 hrs and 35 mins
```

3.Write a Write a C++ program to demonstrate type conversion from a class type to a basic data type using a conversion operator. The program should perform the following tasks:

Class Definition:

Define a class Distance with the following private data members:

- **feet (integer)**
- **inches (float)**

Implement a constructor that takes two arguments: feet and inches, and initializes the respective data members.

Conversion Operator:

Implement a conversion operator in the Distance class that converts the distance (represented by the object) into a basic data type float, representing the total distance in inches.
Main Function:
In the main() function, create an object of the Distance class by providing values for feet and inches.
Convert the Distance object to a float using the conversion operator. This should be done implicitly (i.e., without explicitly calling the conversion function).
Print the total distance in inches.
Use the Display() function to print the time in hours and minutes.

```
#include <iostream>

class Distance {
private:
    int feet;       // Distance in feet
    float inches;   // Distance in inches

public:
    // Constructor that initializes feet and inches
    Distance(int f, float i) : feet(f), inches(i) {}

    // Conversion operator to convert Distance to float (total inches)
    operator float() const {
        return (feet * 12) + inches;  // Convert feet to inches and add
    }

    // Display function to print distance in feet and inches
    void Display() const {
        std::cout << "Distance = " << feet << " feet and " << inches << " inches" <<
std::endl;
    }
};

int main() {
```

```
    // Create a Distance object with feet and inches
    Distance dist(5, 8.5);  // 5 feet and 8.5 inches

    // Implicitly convert the Distance object to float (total inches)
    float totalInches = dist;  // Calls the conversion operator

    // Print the total distance in inches
    std::cout << "Total distance in inches: " << totalInches << " inches" <<
std::endl;

    // Display the distance using the Display function
    dist.Display();

    return 0;
}
```

Output:

```
Total distance in inches: 70.5 inches
Distance: 5 feet and 10.5 inches
```

4.Write Write a C++ program to demonstrate the concept of type conversion from one class type to another using a conversion constructor. The program should perform the following tasks:
Class Definitions:
Define a class Rupees with a public data member amount of type double. Implement a constructor that initializes the amount with a given value.
Define a class Dollars with a public data member amount of type double. Implement a conversion constructor in the Dollars class that takes a Rupees object as an argument and converts the amount from Indian Rupees (INR) to US Dollars (USD). Assume a conversion rate of 1 USD = 83 INR.
Display Functions:
Implement a display() function in both the Rupees and Dollars classes

to print the amount in their respective currencies.
Main Function:
In the main() function, create an object of the Rupees class with an initial amount in INR.
Convert the Rupees object to a Dollars object using the conversion constructor in the Dollars class.
Display the amounts in both INR and USD using the respective display() functions.
Example Output:
Amount in Rupees: 8300 INR
Amount in Dollars: $100

```
#include <iostream>

class Rupees {
public:
    double amount;  // Amount in Rupees

    // Constructor that initializes the amount in Rupees
    Rupees(double a) : amount(a) {}

    // Function to display the amount in Rupees
    void display() const {
        std::cout << "Amount in Rupees: " << amount << " INR" << std::endl;
    }
};

class Dollars {
public:
    double amount;  // Amount in Dollars

    // Conversion constructor that converts Rupees to Dollars
    Dollars(const Rupees & r) {
        amount = r.amount / 83;  // Convert INR to USD using the conversion rate
    }
```

```
    // Function to display the amount in Dollars
    void display() const {
        std::cout << "Amount in Dollars: $" << amount << std::fixed <<
std::setprecision(2) << amount << std::endl;
    }
};

int main() {
    // Create an object of the Rupees class with an initial amount in INR
    Rupees rupeeAmount(8300);  // 8300 INR

    // Convert the Rupees object to a Dollars object using the conversion
constructor
    Dollars dollarAmount(rupeeAmount);

    // Display the amounts in both INR and USD using the respective display
functions
    rupeeAmount.display();  // Display amount in Rupees
    dollarAmount.display();  // Display amount in Dollars

    return 0;
}
```

Output:

```
Amount in Rupees: 8300 INR
Amount in Dollars: $100
```

5.Write a C++ program to demonstrate the conversion of an amount from Indian Rupees (INR) to US Dollars (USD) using a casting operator function. Your program should use two classes: Rupees and Dollars.
Class Definitions:
Class Dollars:
Create a class named Dollars with a public data member amount of type

double.
Implement a constructor that initializes amount with a given value.
Add a display() function that prints the amount in USD in the format: Amount in Dollars: $Y.
Class Rupees:
Create a class named Rupees with a public data member amount of type double.
Implement a constructor that initializes amount with a given value.
Implement a conversion operator function operator Dollars() that converts the amount from INR to USD, assuming a conversion rate of 1 USD = 83 INR.
Add a display() function that prints the amount in INR in the format: Amount in Rupees: X INR.

```
#include <iostream>

class Dollars {
public:
    double amount;  // Amount in Dollars

    // Constructor that initializes amount in Dollars
    Dollars(double a) : amount(a) {}

    // Function to display the amount in Dollars
    void display() const {
        std::cout << "Amount in Dollars: $" << amount << std::fixed <<
std::setprecision(2) << amount << std::endl;
    }
};

class Rupees {
public:
    double amount;  // Amount in Rupees

    // Constructor that initializes amount in Rupees
    Rupees(double a) : amount(a) {}
```

```
    // Conversion operator function to convert Rupees to Dollars
    operator Dollars() const {
        return Dollars(amount / 83);  // Convert INR to USD
    }

    // Function to display the amount in Rupees
    void display() const {
        std::cout << "Amount in Rupees: " << amount << " INR" << std::endl;
    }
};

int main() {
    // Create an object of the Rupees class with an initial amount in INR
    Rupees rupeeAmount(8300);  // 8300 INR

    // Convert the Rupees object to a Dollars object using the conversion operator
    Dollars dollarAmount = rupeeAmount;  // Implicit conversion

    // Display the amounts in both INR and USD using the respective display
functions
    rupeeAmount.display();  // Display amount in Rupees
    dollarAmount.display();  // Display amount in Dollars

    return 0;
}
```

Output:

```
Amount in Rupees: 8300 INR
Amount in Dollars: $100
```