**Slip 7: Sample Solutions and Explanations**

**Q1. C++ Program to Calculate Area and Perimeter of Rectangle**

**Approach**

* Define a Rectangle class with member variables for length and width.
* Provide member functions to calculate area and perimeter.
* Accept input from the user and display the results.

**Code**

#include <iostream>  
using namespace std;  
  
// [Rectangle Class Definition]  
class Rectangle {  
 double length, width;  
public:  
 void accept() {  
 cout << "Enter length: "; cin >> length;  
 cout << "Enter width: "; cin >> width;  
 }  
 double area() { return length \* width; }  
 double perimeter() { return 2 \* (length + width); }  
};  
  
int main() {  
 Rectangle r;  
 r.accept();  
 cout << "Area: " << r.area() << endl;  
 cout << "Perimeter: " << r.perimeter() << endl;  
 return 0;  
}

**Explanation**

* The Rectangle class encapsulates the dimensions and provides methods for area and perimeter.
* The accept method reads values from the user.
* The results are displayed in main.

**Syntax Definitions**

* **class**: A user-defined type that groups data and functions.
* **double**: A data type for floating-point numbers.

**Q2. Class Time with Operator Overloading (!=, >>, <<)**

**Approach**

* Define a Time class with hours, minutes, and seconds.
* Overload the != operator to compare two Time objects.
* Overload the >> and << operators for input and output.

**Code**

#include <iostream>  
using namespace std;  
  
// [Time Class Definition]  
class Time {  
 int h, m, s;  
public:  
 // [Input Operator >>]  
 friend istream& operator>>(istream& in, Time& T) {  
 cout << "Hours: "; in >> T.h;  
 cout << "Minutes: "; in >> T.m;  
 cout << "Seconds: "; in >> T.s;  
 return in;  
 }  
 // [Output Operator <<]  
 friend ostream& operator<<(ostream& out, const Time& T) {  
 out << T.h << ":" << T.m << ":" << T.s;  
 return out;  
 }  
 // [Inequality Operator !=]  
 bool operator!=(const Time& t) {  
 return h != t.h || m != t.m || s != t.s;  
 }  
};  
  
int main() {  
 Time t1, t2;  
 cin >> t1;  
 cin >> t2;  
 cout << "Time 1: " << t1 << endl;  
 cout << "Time 2: " << t2 << endl;  
 if (t1 != t2)  
 cout << "Times are not equal." << endl;  
 else  
 cout << "Times are equal." << endl;  
 return 0;  
}

**Explanation**

* The Time class uses friend functions to overload input and output operators.
* The != operator checks if any component of the time differs.
* Demonstrates operator overloading for user-defined types.

**Syntax Definitions**

* **friend**: Allows a function to access private/protected members of a class.
* **operator overloading**: Redefining operators for user-defined types.
* **istream/ostream**: Standard input/output stream classes in C++.

**Q3. Financial Application: Money Class with Operator Overloading**

**Approach**

* Define a Money class with rupees and paise.
* Overload the + operator to add two Money objects, normalizing paise to rupees.
* Provide a display method.

**Code**

#include <iostream>  
using namespace std;  
  
// [Money Class Definition]  
class Money {  
 int rupees, paise;  
public:  
 Money(int r=0, int p=0): rupees(r), paise(p) { normalize(); }  
 void normalize() { rupees += paise/100; paise %= 100; }  
 Money operator+(const Money& m) { return Money(rupees + m.rupees, paise + m.paise); }  
 void display() { cout << rupees << " Rupees " << paise << " Paise\n"; }  
};  
  
int main() {  
 Money m1(10,150), m2(3,95), m3;  
 m3 = m1 + m2;  
 m3.display();  
 return 0;  
}

**Explanation**

* The Money class handles currency addition and normalization (e.g., 150 paise = 1 rupee 50 paise).
* The + operator is overloaded to add two Money objects.
* The display method prints the result in a readable format.

**Syntax Definitions**

* **operator+**: Overloads the + operator for user-defined types.
* **Constructor with default arguments**: Allows object creation with or without parameters.
* **Normalization**: Adjusts paise to rupees if paise >= 100.