**Slip 25: Sample Solutions and Explanations**

**Q1. Function Overloading: Calculate Area of Square and Rectangle**

**Approach**

* Use function overloading to define two area functions: one for a square (one argument), one for a rectangle (two arguments).
* Each function returns the area using the appropriate formula.
* Demonstrate both in main.

**Code**

#include <iostream>  
using namespace std;  
  
// [Area of Square]  
int area(int side) { return side \* side; }  
// [Area of Rectangle]  
int area(int l, int b) { return l \* b; }  
  
int main() {  
 cout << "Area of square (4): " << area(4) << endl;  
 cout << "Area of rectangle (4,6): " << area(4,6) << endl;  
 return 0;  
}

**Explanation**

* Two area functions are defined: one takes one argument (square), one takes two (rectangle).
* The compiler chooses the correct function based on the number of arguments.
* Demonstrates function overloading for different shapes.

**Syntax Definitions**

* **Function Overloading**: Defining multiple functions with the same name but different parameter lists.

**Q2. Student Class: Accept, Display, and Search by Area (for Hostel Allocation)**

**Approach**

* Create a Student class with attributes: roll number, name, and area (e.g., "North", "South").
* Accept details for n students and store them in a vector.
* Display details of students from a specific area.

**Code**

#include <iostream>  
#include <vector>  
using namespace std;  
// [Student Class Definition]  
class Student {  
 int roll;  
 string name, area;  
public:  
 void accept() {  
 cout << "Roll: "; cin >> roll;  
 cout << "Name: "; cin >> name;  
 cout << "Area: "; cin >> area;  
 }  
 void display() { cout << roll << " " << name << " " << area << endl; }  
 string getArea() { return area; }  
};  
  
int main() {  
 int n;  
 cout << "Number of students: ";  
 cin >> n;  
 vector<Student> students(n);  
 for(auto &s : students) s.accept();  
 string searchArea;  
 cout << "Enter area to search: "; cin >> searchArea;  
 cout << "Students from area " << searchArea << ":\n";  
 for(auto &s : students)  
 if(s.getArea() == searchArea) s.display();  
 return 0;  
}

**Explanation**

* The Student class encapsulates student data and provides methods to accept and display it.
* The program reads n students, then displays those from the specified area.
* The getArea method is used for filtering.

**Syntax Definitions**

* **class**: A user-defined type that groups data and functions.
* **vector**: A dynamic array from the C++ Standard Library.

**Q3. Hostel Allocation System (Case Study)**

**Approach**

* Use a Student class with roll, name, and area.
* Accept, display, and search for students by area for hostel allocation.
* Demonstrate object usage and filtering in main.

**Code**

#include <iostream>  
#include <vector>  
using namespace std;  
class Student {  
 int roll;  
 string name, area;  
public:  
 void accept() { cin >> roll >> name >> area; }  
 void display() { cout << roll << " " << name << " " << area << endl; }  
 string getArea() { return area; }  
};  
  
int main() {  
 int n; cin >> n;  
 vector<Student> students(n);  
 for(auto &s : students) s.accept();  
 string area; cin >> area;  
 for(auto &s : students)  
 if(s.getArea() == area) s.display();  
 return 0;  
}

**Explanation**

* Accepts and displays students, filters by area for hostel allocation.
* Demonstrates object usage and filtering for a real-world scenario.

**Syntax Definitions**

* **class**: A user-defined type that groups data and functions.
* **vector**: A dynamic array from the C++ Standard Library.