**Slip 15: Sample Solutions and Explanations**

**Q1. Function Overloading: Find Maximum of Two and Three Numbers**

**Approach**

* Use function overloading to define two max functions: one for two arguments, one for three arguments.
* Each function returns the largest value among its parameters.
* Demonstrate both in main.

**Code**

#include <iostream>  
using namespace std;  
  
// [Max of Two]  
int max(int a, int b) { return (a > b) ? a : b; }  
// [Max of Three]  
int max(int a, int b, int c) { return (a > b && a > c) ? a : (b > c ? b : c); }  
  
int main() {  
 cout << "Max of 5, 9: " << max(5, 9) << endl;  
 cout << "Max of 3, 7, 2: " << max(3, 7, 2) << endl;  
 return 0;  
}

**Explanation**

* Two max functions are defined: one for two integers, one for three.
* The compiler chooses the correct function based on the number of arguments.
* The ternary operator is used for concise comparison.

**Syntax Definitions**

* **Function Overloading**: Defining multiple functions with the same name but different parameter lists.
* **Ternary Operator**: A shorthand for if-else: (condition) ? value\_if\_true : value\_if\_false.

**Q2. Student Class: Accept, Display, and Search by Roll Number**

**Approach**

* Create a Student class with attributes: roll number, name, and marks.
* Accept details for n students and store them in a vector.
* Search for a student by roll number and display their details.

**Code**

#include <iostream>  
#include <vector>  
using namespace std;  
  
// [Student Class Definition]  
class Student {  
 int roll;  
 string name;  
 double marks;  
public:  
 void accept() {  
 cout << "Roll: "; cin >> roll;  
 cout << "Name: "; cin >> name;  
 cout << "Marks: "; cin >> marks;  
 }  
 void display() { cout << roll << " " << name << " " << marks << endl; }  
 int getRoll() { return roll; }  
};  
  
int main() {  
 int n;  
 cout << "Number of students: ";  
 cin >> n;  
 vector<Student> students(n);  
 for(auto &s : students) s.accept();  
 int searchRoll;  
 cout << "Enter roll to search: "; cin >> searchRoll;  
 bool found = false;  
 for(auto &s : students) {  
 if(s.getRoll() == searchRoll) {  
 s.display();  
 found = true;  
 }  
 }  
 if(!found) cout << "Student not found." << endl;  
 return 0;  
}

**Explanation**

* The Student class encapsulates student data and provides methods to accept and display it.
* The program reads n students, then searches for a student by roll number and displays their details if found.
* The getRoll method is used for searching.

**Syntax Definitions**

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

**Q3. Account Management System (Static Count & Dynamic Array Case Study)**

**Approach**

* Use a class Account with static members to count and sum balances.
* Store accounts in a dynamic array/vector.
* Accept, display, deposit, and withdraw with validation.

**Code**

#include <iostream>  
#include <vector>  
using namespace std;  
  
class Account {  
 static int totalAccounts;  
 static double totalBalance;  
 int accNum;  
 double balance;  
public:  
 Account() { totalAccounts++; }  
 void accept() {  
 cout << "Acc No: "; cin >> accNum;  
 cout << "Balance: "; cin >> balance;  
 totalBalance += balance;  
 }  
 void display() { cout << accNum << " " << balance << endl; }  
 void deposit(double amt) { balance += amt; totalBalance += amt; }  
 void withdraw(double amt) {  
 if(balance >= amt) { balance -= amt; totalBalance -= amt; }  
 else cout << "Insufficient balance\n";  
 }  
 static void stats() {  
 cout << "Total Accounts: " << totalAccounts << ", Total Balance: " << totalBalance << endl;  
 }  
};  
int Account::totalAccounts = 0;  
double Account::totalBalance = 0.0;  
  
int main() {  
 int n; cout << "Number of accounts: "; cin >> n;  
 vector<Account> accs(n);  
 for(auto &a : accs) a.accept();  
 for(auto &a : accs) a.display();  
 accs[0].deposit(500); accs[1].withdraw(100);  
 Account::stats();  
 return 0;  
}

**Explanation**

* Static members totalAccounts and totalBalance track the number of accounts and total balance.
* Methods allow for deposit, withdrawal, and displaying statistics.
* Demonstrates static data members and dynamic arrays.

**Syntax Definitions**

* **static**: Declares a member function or variable that belongs to the class, not to any object.
* **vector**: A dynamic array from the C++ Standard Library.
* **Dynamic Array**: An array whose size can change at runtime (here, implemented using vector).