**Slip 5: Sample Solutions and Explanations**

**Q1. Function Overloading: Volume of Cone and Sphere**

**Approach**

* Use function overloading to define two volume functions: one for a cone (radius and height), one for a sphere (radius only).
* Calculate the volume using the respective formulas and demonstrate both in main.

**Code**

#include <iostream>  
using namespace std;  
  
// [Volume of Cone]  
double volume(double r, double h) {  
 return (3.14159 \* r \* r \* h) / 3;  
}  
// [Volume of Sphere]  
double volume(double r) {  
 return (4.0 / 3.0) \* 3.14159 \* r \* r \* r;  
}  
int main() {  
 cout << "Volume of cone (r=2, h=4): " << volume(2, 4) << endl;  
 cout << "Volume of sphere (r=2): " << volume(2) << endl;  
 return 0;  
}

**Explanation**

* Two volume functions are defined: one takes two arguments (cone), one takes one (sphere).
* The compiler chooses the correct function based on the number of arguments.
* The formulas for volume are directly implemented.

**Syntax Definitions**

* **Function Overloading**: Defining multiple functions with the same name but different parameter lists.
* **double**: A data type for floating-point numbers.

**Q2. Template Vector Class (Create, Modify, Scalar Multiply, Display)**

**Approach**

* Create a template class for a vector of type T.
* Implement methods to create, modify, multiply by a scalar, and display the vector.

**Code**

#include <iostream>  
#include <vector>  
using namespace std;  
  
template <typename T>  
class MyVector {  
 vector<T> data;  
public:  
 // [Create Vector]  
 void create(int n) {  
 cout << "Enter " << n << " elements:\n";  
 T val;  
 for (int i = 0; i < n; ++i) {  
 cin >> val; data.push\_back(val);  
 }  
 }  
 // [Modify Element]  
 void modify(int idx, T val) {  
 if (idx >= 0 && idx < data.size()) data[idx] = val;  
 }  
 // [Scalar Multiplication]  
 void multiply(T scalar) {  
 for (auto &v : data) v \*= scalar;  
 }  
 // [Display Vector]  
 void display() {  
 cout << "(";  
 for (size\_t i = 0; i < data.size(); ++i) {  
 cout << data[i];  
 if (i < data.size() - 1) cout << ",";  
 }  
 cout << ")\n";  
 }  
};  
  
int main() {  
 MyVector<int> v;  
 v.create(3);  
 v.display();  
 v.modify(1, 99); v.display();  
 v.multiply(2); v.display();  
 return 0;  
}

**Explanation**

* The template class MyVector can store any type of vector.
* Methods allow creation, modification, scalar multiplication, and display of the vector.
* Demonstrated with int type in main.

**Syntax Definitions**

* **template <typename T>**: Allows the class to work with any data type.
* **vector**: A dynamic array from the C++ Standard Library.
* **auto**: Automatically deduces the type of the variable from its initializer.

**Q3. Math Utility Library (Static Functions Case Study)**

**Approach**

* Create a class with static functions for factorial and prime checking.
* Call these functions directly using the class name, without creating an object.

**Code**

#include <iostream>  
using namespace std;  
  
class MathLib {  
public:  
 // [Static Factorial Function]  
 static int factorial(int n) {  
 int result = 1;  
 for(int i = 1; i <= n; ++i) result \*= i;  
 return result;  
 }  
 // [Static Prime Check Function]  
 static bool isPrime(int n) {  
 if(n < 2) return false;  
 for(int i = 2; i\*i <= n; ++i)  
 if(n % i == 0) return false;  
 return true;  
 }  
};  
  
int main() {  
 int x;  
 cout << "Enter number for factorial: "; cin >> x;  
 cout << "Factorial: " << MathLib::factorial(x) << endl;  
 cout << "Enter number for prime check: "; cin >> x;  
 cout << (MathLib::isPrime(x) ? "Prime" : "Not Prime") << endl;  
 return 0;  
}

**Explanation**

* The MathLib class provides static methods for factorial and prime checking.
* Static methods can be called using the class name without creating an object.
* The program demonstrates both functionalities with user input.

**Syntax Definitions**

* **static**: Declares a member function or variable that belongs to the class, not to any object.
* **bool**: A data type that can hold true or false.
* **?: (ternary operator)**: A shorthand for if-else to choose between two values.