**Slip 8: Sample Solutions and Explanations**

**Q1. Function Overloading: Swap for Int and Double**

**Approach**

* Define two overloaded swap functions: one for int and one for double.
* Use reference parameters to swap values in place.
* Demonstrate both swaps in main.

**Code**

#include <iostream>  
using namespace std;  
  
// [Swap Ints]  
void swap(int &a, int &b) {  
 int t = a; a = b; b = t;  
}  
// [Swap Doubles]  
void swap(double &a, double &b) {  
 double t = a; a = b; b = t;  
}  
int main() {  
 int i1 = 5, i2 = 9;  
 double d1 = 2.5, d2 = 8.7;  
 swap(i1, i2); swap(d1, d2);  
 cout << "Swapped ints: " << i1 << " " << i2 << endl;  
 cout << "Swapped doubles: " << d1 << " " << d2 << endl;  
 return 0;  
}

**Explanation**

* Two swap functions are defined, one for int and one for double.
* Reference parameters allow the function to modify the original variables.
* Demonstrates function overloading and in-place swapping for both types.

**Syntax Definitions**

* **Function Overloading**: Defining multiple functions with the same name but different parameter types.
* **Reference Parameter**: Allows a function to modify the caller's variable directly.

**Q2. Mobile Class with Accept/Sale/Purchase**

**Approach**

* Create a Mobile class with attributes: company, model, color, price, quantity.
* Implement methods to accept details, sell (decrement quantity), purchase (increment quantity), and display.

**Code**

#include <iostream>  
using namespace std;  
  
// [Mobile Class Definition]  
class Mobile {  
 string company, model, color;  
 double price;  
 int quantity;  
public:  
 void accept() {  
 cout << "Company: "; cin >> company;  
 cout << "Model: "; cin >> model;  
 cout << "Color: "; cin >> color;  
 cout << "Price: "; cin >> price;  
 cout << "Quantity: "; cin >> quantity;  
 }  
 void sale(int qty) {  
 if (quantity >= qty) {  
 quantity -= qty;  
 cout << "Sale success. Remaining: " << quantity << endl;  
 } else cout << "Not enough stock.\n";  
 }  
 void purchase(int qty) {  
 quantity += qty;  
 cout << "Purchased. Total stock: " << quantity << endl;  
 }  
 void display() {  
 cout << company << " " << model << " " << color << " " << price << " " << quantity << endl;  
 }  
};  
  
int main() {  
 Mobile m;  
 m.accept();  
 m.display();  
 m.sale(3); // Sale demo  
 m.purchase(5); // Purchase demo  
 m.display();  
 return 0;  
}

**Explanation**

* The Mobile class manages product details and stock operations.
* The sale method checks for sufficient stock before selling.
* The purchase method adds to the stock.
* The display method prints all details.

**Syntax Definitions**

* **class**: A user-defined type that groups data and functions.
* **Method**: A function defined inside a class.

**Q3. Bookshop Management (Inventory & Sale Case Study)**

**Approach**

* Define a Book class with attributes: author, title, price, publisher, stock.
* Accept book details, search by title/author, check stock, and print bill if available.

**Code**

#include <iostream>  
#include <vector>  
using namespace std;  
  
// [Book Class Definition]  
class Book {  
 string author, title, publisher;  
 double price;  
 int stock;  
public:  
 void accept() {  
 cout << "Title: "; cin >> title;  
 cout << "Author: "; cin >> author;  
 cout << "Publisher: "; cin >> publisher;  
 cout << "Price: "; cin >> price;  
 cout << "Stock: "; cin >> stock;  
 }  
 bool match(string t, string a) { return title==t && author==a; }  
 bool available(int qty) { return stock >= qty; }  
 double bill(int qty) { return price \* qty; }  
 void sell(int qty) { stock -= qty; }  
 void display() {  
 cout << title << " " << author << " " << publisher << " " << price << " " << stock << endl;  
 }  
};  
  
int main() {  
 int n; cout << "Books? "; cin >> n;  
 vector<Book> shop(n);  
 for(auto &b:shop) b.accept();  
 string reqTitle, reqAuthor; int reqQty;  
 cout << "Title/Author/QTY: "; cin >> reqTitle >> reqAuthor >> reqQty;  
 for(auto &b:shop) {  
 if(b.match(reqTitle, reqAuthor)) {  
 if(b.available(reqQty)) {  
 cout << "Total Cost: " << b.bill(reqQty) << endl;  
 b.sell(reqQty);  
 } else cout << "Insufficient stock\n";  
 }  
 }  
 return 0;  
}

**Explanation**

* The Book class manages inventory and sales operations.
* The program accepts book details, searches for a match, checks stock, and processes the sale.
* The bill is calculated and stock updated if the sale is successful.

**Syntax Definitions**

* **vector**: A dynamic array from the C++ Standard Library.
* **Method**: A function defined inside a class.