**Slip 12: Sample Solutions and Explanations**

**Q1. Swap Two Numbers Using Reference Variables**

**Approach**

* Define a function that takes two integer references as parameters.
* Swap their values inside the function using a temporary variable.
* Call the function from main and display the numbers before and after swapping.

**Code**

#include <iostream>  
using namespace std;  
  
// [Swap Function Definition]  
void swap(int &a, int &b) {  
 int temp = a;  
 a = b;  
 b = temp;  
}  
  
int main() {  
 int x, y;  
 cout << "Enter two numbers: ";  
 cin >> x >> y;  
 cout << "Before swap: x=" << x << " y=" << y << endl;  
 swap(x, y);  
 cout << "After swap: x=" << x << " y=" << y << endl;  
 return 0;  
}

**Explanation**

* The swap function uses reference parameters (int &a, int &b), so changes inside the function affect the original variables in main.
* A temporary variable is used to hold one value during the swap.
* The program demonstrates the effect by printing values before and after the swap.

**Syntax Definitions**

* **Reference Parameter (int &a)**: Allows the function to modify the original variable passed from the caller.

**Q2. Customer Class: Accept/Display/Filter by Name Starting with 'P'**

**Approach**

* Create a Customer class with attributes: code, name, and population.
* Accept details for n customers and store them in a vector.
* Display details of customers whose names start with the letter 'P'.

**Code**

#include <iostream>  
#include <vector>  
using namespace std;  
  
// [Class Definition]  
class Customer {  
 int code, population;  
 string name;  
public:  
 void accept() {  
 cout << "Code: "; cin >> code;  
 cout << "Name: "; cin >> name;  
 cout << "Population: "; cin >> population;  
 }  
 void display() { cout << code << " " << name << " " << population << endl; }  
 string getName() { return name; }  
};  
  
int main() {  
 int n;  
 cout << "Number of customers: ";  
 cin >> n;  
 vector<Customer> customers(n);  
 for(auto &c : customers) c.accept();  
 cout << "Customers with names starting with 'P':\n";  
 for(auto &c : customers)  
 if(!c.getName().empty() && c.getName()[0] == 'P')  
 c.display();  
 return 0;  
}

**Explanation**

* The Customer class encapsulates customer data and provides methods to accept and display it.
* The program reads n customers, then iterates through the list, displaying only those whose names start with 'P'.
* The check !c.getName().empty() ensures the name is not empty before accessing the first character.

**Syntax Definitions**

* **class**: A user-defined type that groups data and functions.
* **vector**: A dynamic array from the C++ Standard Library that can grow or shrink in size.

**Q3. Library Management System (Case Study)**

**Approach**

* Define a Book class with attributes: title, author, year, and ISBN.
* Accept details for multiple books and store them in a vector.
* Allow the user to search for a book by ISBN and display its details if found.

**Code**

#include <iostream>  
#include <vector>  
using namespace std;  
  
// [Book Class Definition]  
class Book {  
 string title, author, isbn;  
 int year;  
public:  
 void accept() {  
 cout << "Title: "; cin >> title;  
 cout << "Author: "; cin >> author;  
 cout << "Year: "; cin >> year;  
 cout << "ISBN: "; cin >> isbn;  
 }  
 void display() {  
 cout << title << " " << author << " " << year << " " << isbn << endl;  
 }  
 string getISBN() { return isbn; }  
};  
  
int main() {  
 int n;  
 cout << "Number of books: ";  
 cin >> n;  
 vector<Book> library(n);  
 for(auto &b : library) b.accept();  
 string searchISBN;  
 cout << "Enter ISBN to search: ";  
 cin >> searchISBN;  
 bool found = false;  
 for(auto &b : library) {  
 if(b.getISBN() == searchISBN) {  
 b.display();  
 found = true;  
 }  
 }  
 if(!found) cout << "Book not found." << endl;  
 return 0;  
}

**Explanation**

* The Book class encapsulates book details and provides methods for input and output.
* The program reads n books, then searches for a book by ISBN and displays its details if found.
* The found flag ensures a message is shown if no book matches the search.

**Syntax Definitions**

* **class**: Encapsulates data and functions for a specific concept (here, a book).
* **vector**: A resizable array from the C++ Standard Library.
* **Method**: A function defined inside a class.