**Slip 16: Sample Solutions and Explanations**

**Q1. Exception Handling: Validate Age, Income, City, Vehicle**

**Approach**

* Take user input for age, income, city, and vehicle type.
* Validate each input against specific constraints:
  + Age: 18–55
  + Income: 50,000–100,000
  + City: Pune, Mumbai, Bangalore, Chennai
  + Vehicle: Must be "4Wheeler"
* Throw and catch exceptions for each invalid input, displaying a specific error message.

**Code**

#include <iostream>  
#include <stdexcept>  
using namespace std;  
  
// [Validation Function]  
void validate(int age, double income, string city, string vehicle) {  
 if(age < 18 || age > 55) throw runtime\_error("AGE ERROR");  
 if(income < 50000 || income > 100000) throw runtime\_error("INCOME ERROR");  
 if(city != "Pune" && city != "Mumbai" && city != "Bangalore" && city != "Chennai")  
 throw runtime\_error("CITY ERROR");  
 if(vehicle != "4Wheeler") throw runtime\_error("VEHICLE ERROR");  
}  
  
int main() {  
 int age; double income; string city, vehicle;  
 cout << "Age, income, city, vehicle: ";  
 cin >> age >> income >> city >> vehicle;  
 try {  
 validate(age, income, city, vehicle);  
 cout << "All constraints satisfied!\n";  
 } catch(const runtime\_error& e) {  
 cout << "Exception: " << e.what() << endl;  
 }  
 return 0;  
}

**Explanation**

* The validate function checks each constraint and throws a runtime\_error with a specific message if a rule is violated.
* The main function catches the exception and prints the error message.
* Demonstrates exception handling for multiple business rules.

**Syntax Definitions**

* **throw**: Used to signal the occurrence of an exception.
* **try-catch**: Used to handle exceptions and prevent program termination.
* **runtime\_error**: A standard exception class for runtime errors.

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

**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 name 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; }  
 string getName() { return name; }  
};  
  
int main() {  
 int n;  
 cout << "Number of students: ";  
 cin >> n;  
 vector<Student> students(n);  
 for(auto &s : students) s.accept();  
 string searchName;  
 cout << "Enter name to search: "; cin >> searchName;  
 bool found = false;  
 for(auto &s : students) {  
 if(s.getName() == searchName) {  
 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 name and displays their details if found.
* The getName 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. Exception Handling with Multiple Constraints (Case Study)**

**Approach**

* Take user input for age, income, city, and vehicle.
* Validate all constraints; throw/catch exceptions showing specific messages for each condition.
* Demonstrate robust exception handling for business logic.

**Code**

#include <iostream>  
#include <stdexcept>  
using namespace std;  
  
void validate(int age, double income, string city, string vehicle) {  
 if(age < 18 || age > 55) throw runtime\_error("AGE ERROR");  
 if(income < 50000 || income > 100000) throw runtime\_error("INCOME ERROR");  
 if(city != "Pune" && city != "Mumbai" && city != "Bangalore" && city != "Chennai")  
 throw runtime\_error("CITY ERROR");  
 if(vehicle != "4Wheeler") throw runtime\_error("VEHICLE ERROR");  
}  
  
int main() {  
 int age; double income; string city, vehicle;  
 cout << "Enter age, income, city, vehicle: ";  
 cin >> age >> income >> city >> vehicle;  
 try {  
 validate(age, income, city, vehicle);  
 cout << "All constraints satisfied!\n";  
 } catch(const runtime\_error& e) {  
 cout << "Exception: " << e.what() << endl;  
 }  
 return 0;  
}

**Explanation**

* The validate function checks each constraint and throws a runtime\_error with a specific message if a rule is violated.
* The main function catches the exception and prints the error message.
* Demonstrates exception handling for multiple business rules.

**Syntax Definitions**

* **throw**: Used to signal the occurrence of an exception.
* **try-catch**: Used to handle exceptions and prevent program termination.
* **runtime\_error**: A standard exception class for runtime errors.