CASE STUDY & PROBLEM STATEMNT

Title: Advanced C++ (Kaizen)

KPIT Technologies Ltd.

Document Revision History

Version	Date	Author (s)	Reviewer (s)	Description
1.0	16th August, 2019	Rajesh Sola		Initial draft
2.0				
3.0				

Contents

Assessment Type	Marks weightage to module	Submission deadline	Contribution
Open book case study	100%	As per mailer	Individual

Submission package:

- Source code for given problem statement
- Class diagram (Hand written / Generated by any UML tool)
- Considerations on problem solution (Embedded comments or any simple format)
- Optional – a small report on your C++ learning experience (Any simple format – simple text/Mark Down/Excel/Slides)
- Please zip the artifacts for mailing purpose.
- Optional – You can host your code in a git repo under gitlab.kpit.com, add developer access to module leader (Rajesh Sola).

Learning Outcomes:

Learning outcome	
O1	Applied Object Oriented concepts
O2	Effective & Clean coding practices of C++
O3	Usage of Modern C++ Features
O4	Code Quality Metrics
O5	Simple Test Driven Approach

Case Study Description

Part-I: OO Design

Analyze the problem statement and identify suitable classes and appropriate relationships between the classes.

Part-II: C++ Features

a) Prepare a basic solution to meet all the expected functionality as per the given problem statement.
b) Clean & effective coding paradigms
c) Provide crisp logic, avoid redundant logic & length functions. Consider Single Responsibility Principle (SRP).
d) Isolate console i/o, file operations from Business classes
e) Use std algorithms wherever possible instead of your own iterator based logic

Part -III: Modern C++ Features

a) Apply possible C++11 & 14 features. Here is a sample listing. You may consider maximum, if not all. You may add any other not listed also.
    a. New additions on language basics, classes & objects, inheritance, templates etc.
    b. Move semantics
    c. Lambda expressions, std::function, std::bind
    d. STL improvements in C++11/14
    e. Smart pointers
    f. Chrono library, Threading & IPC (If possible)

Part-IV: Code Quality Factors

a. Adhere to code Style, for e.g. Google Style
b. No violations from Static Analysis, e.g. cppcheck
c. No memory leaks / Heap issues
d. Meaningful names

Part-V: Simple Testing Strategy

a. Suitable code in main to cover all functionality
b. Basic asserts (if possible)

-------------------------------End of guidelines---------------------------------

Problem Statement:-

In this problem you are supposed to design a solution for Flight Management as described below. Let's not consider date & time of flight journey to make the problem simpler for now. Let's assume flight number is unique in entire database

- Design a class Flight as follows
    - Attributes
        - Flight number
        - Origin City
        - Destination city
        - Operator (AirIndia, Indigo etc)
        - Air Fare
    - Suitable constructor(s)
    - Any additional member functions as required
- Design another class to maintain Flight database
    - Use suitable container(s) to store all flight details
    - Constructor, Destructor if required
    - Operations
        - AddTrip with given attributes
        - RemoveTrip by flight number
        - Update air fare by flight number
        - Find flight details by number
        - Find all flights departing from a particular city
        - Find all flights by a specific operator
        - Find average air fare from all trips
        - Find minimum fare between two cities
        - Find maximum fare by a particular operator
        - Update the air fare for all flights by a specific operator, say decrease by 10%

A basic template is attached along with this code, you are free to use any other classes, global functions to meet the mentioned functionality and guidelines.

---------------------------------- **THANK YOU** ----------------------------------