

Course Code: CS-1004	Course Name: Object Oriented Programming
Instructor Name / Names: Ms. Atiya Jokhio	
Section-BDS	Student-ID:

Time Allowed: 30 minutes.

Total Points: 10

TYPE B

Question:1 Complete the missing lines/line of code

[5 points]

```
#include <iostream>
#include <string>
//include missing header file here
#include <fstream>

// Function to display the content of a file
void displayFileContent(const string & filename) {
    ifstream file(filename);
    string line;

    /* Check if the file was successfully opened and Read and Display each line from the file and then
    close the file

    if (file.is_open()) { //
        std::cout << "File content:" << std::endl; // Displaying a message indicating file content
        while (std::getline(file, line)) { //
            std::cout << line << std::endl; // Display each line of the file
        }
        file.close(); // Close the file
    } else {
        std::cout << "Failed to open the file." << std::endl; // Display an error message if file opening
        failed
    }
}

int main() {
    displayFileContent("new_test.txt"); // Display content of "new_test.txt" before any modification
    cout << endl;

    ofstream outputFile;
    // Open the file in append mode
    outputFile.open("new_test.txt", std::ios::app); // Open "new_test.txt" in append mode
```

```

displayFileContent("new_test.txt"); // Display content of "new_test.txt" after opening in append
mode
cout << endl;

if (outputFile.is_open()) { // Check if the file was successfully opened
    string newData; // Declare a string to store new data entered by the user

    cout << "Enter the data to append: "; // Prompt the user to enter data
    // Read the new data from the user
    getline(cin, newData); // Get user input for new data

    // Append the new data to the file
    outputFile << newData << endl; // Write the new data to the file
    outputFile.close(); // Close the file
}

```

Question:2

[5 points]

Design an abstract base class `Passenger` with a function `displayType()` that shows the type of passenger (implemented inside the base class) and a **pure virtual function** `bookTicket()` to be overridden by all derived classes.

- Create three derived classes: Each class must implement its specific version of `bookTicket()` and their own specific behaviors.
 - **RegularPassenger**
 - **FrequentFlyer**
 - **VIPPassenger**:

You need to **Demonstrating Polymorphism** by storing different types of passengers in an array of `Passenger*` and calling the respective methods on each passenger.