National University of Computer & Emerging Sciences, Karachi
Spring-2025 School of Computing
Quiz No. 3
8th May 2025

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

Time Allowed: 30 minutes. **Total Points**: 10

TYPE A

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 User for an online shopping system. Which includes a **concrete method** displayUserType() that prints the type of user (this function is implemented inside the base class) and a **pure virtual method** browseItems() which is to be overridden in all derived classes.
Create three derived classes: Each class must provide its specific version of browseItems().

1. **GuestUser**
2. **RegisteredUser**
3. **PremiumUser**

Demonstrate runtime polymorphism by:

- Creating an array of User*. *(Hint: User* users[numUsers];)*
- Storing different types of users in the container.
- Looping through the users and calling both displayUserType() and browseItems() using only User* pointers.
- Ensure proper memory management by deleting all dynamically allocated objects at the end of the program.