National University of Computer & Emerging Sciences, Karachi
Spring-2025, School of Computing (BSCS, BSSE, BSAI, BSCY)

Assignment # 3

Subject: Object Oriented Programming **Total Marks: 40**	**Post Date: 22nd April 2025** **Due Date: 2nd May 2025**
Course Instructors: Ms. Abeeha Sattar.	

Instructions to be strictly followed.

1. **Each student should submit these files:**
    a. **A zip of all source files named as "A3-Q#[StudentID]" where # is the question number and Student ID is your ID.**
    b. **A DOC file where they copy code for each question and screen shot of the output. This document contains all the questions, answer codes and output in sequence. Name this document as "A3-[StudentID].docx".**
    c. **All the submissions will be made on Google Classroom.**
2. **Each output should have STUDENT ID and NAME of the student at the top.**
3. **It should be clear that your assignment would not get any credit if the assignment is submitted after the due date.**
4. **Zero grade for plagiarism (copy/ cheating) and late submissions.**

Scenario # 1:

The one and only task you'll be doing this time!

Once more, we'll be picking up where we left off in the last assignment with **FAST's Transportation System**.

For this scenario, you are required to revisit the system, and update it according to the newer concepts that we have discussed. It should now also include the following concepts: Abstract Classes, Exception Handling, File Handling, Templates (if we manage to complete it in class before assignment is due, otherwise you can leave this out)

Things to consider:

You already have a lot of progress from assignment # 1 and assignment # 2 regarding this question. You should have majority of the functionality down that is required by your system to function properly.

Observe the classes you've created, if there's a parent-child relationship between those classes, ask yourself this: "*Is the object of the parent class required? If not, can I convert it into an abstract class to generalize things?*" For example, you can make the "user" class abstract with a pure virtual function for login. The students, teachers and admins (all children of the user class) can then override this function to login differently – let's say one logs in with a password, other with an OTP, etc. Furthermore, you can generalize the login function by having just one "user" pointer, which can access the login functionality (and other functionalities too, if you have identified them). Now this is just one example to help you get thinking—look at other functions, can they be made pure virtual? Is there a need for them to be pure virtual? If so, go ahead and do it!

Moving on, you are now required to store the data of the system into files as well. Some suggestions regarding this:

1. Store data about each class separately, i.e., in different files. Admin info can go into *admin.bin*, points info in *points.bin*, etc.
2. At the top of each file keep the number of records as an integer. This way you will know exactly how much information you need to read from the file when reading it.
3. When your program starts up, have a routine (function) which reads from the relevant files and stores information into the required variables. This might not be necessary if your program doesn't need data right from the start. In which case, see point 4.
4. Whenever some information about some types of objects is needed, then and only then read from the file. For example, no need to fetch all the points from *points.bin* until an admin wants to look at the list of points.

5. Once you have read information from the file, no need to keep updating it right away. You can choose to save all the information when: a function ends, or when the program exits. Do not that storing everything back to files before the program ends will cause a lot of work for your system, so try not to do all of it at the end.

Some notes for exception handling:

You're bound to run into memory management issues or file handling issues. For file handling, feel free to use error prevention techniques that we have discussed in class, using the *fail()* and *bad()* functions associated with the stream object. For memory management purposes, use exception handling.

Keeping all of these things in mind, show what your updated class diagram looks and update your previous code accordingly.

Create an additional document which contains the walkthrough from the users' (student, teacher, admin) perspectives. It should highlight what actions are available to the user at any given point, and what happens when a specific action is taken. This walkthrough is not meant to be technical.

Here's an example of a user ordering food using an application to give you an idea; courtesy of ChatGPT :"D

🍔 **Food Ordering App – User Walkthrough**

1. Launching the App

- The user taps on the food ordering app icon.
- A splash screen with the app logo appears, followed by the home screen showing featured restaurants and categories like "Pizza," "Burgers," and "Desserts."

2. Browsing Restaurants

- The user taps on the "Burgers" category.
- A list of burger restaurants nearby is displayed with ratings, delivery time, and estimated cost.
- The user selects "Big Bite Burgers."

3. Viewing the Menu

- The restaurant page opens, showing the full menu with categories: "Combos," "Burgers," "Fries," "Drinks."
- The user scrolls and taps on a "Cheeseburger Combo."

4. Customizing the Order

- A pop-up allows customization: cheese type, bun preference, add-ons (e.g., bacon, jalapeños).
- The user selects “Cheddar,” adds “bacon,” and chooses “curly fries” as a side.
- Taps “Add to Cart.”

5. Reviewing the Cart

- The user opens the cart icon.
- Sees the Cheeseburger Combo, subtotal, taxes, and delivery fee.
- Applies a promo code for 10% off.

6. Placing the Order

- Taps “Proceed to Checkout.”
- Confirms the delivery address and payment method.
- Taps “Place Order.”

7. Tracking the Delivery

- The app shows an order confirmation screen and estimated delivery time.
- The user can track the driver’s location in real-time on a map.

8. Order Delivered

- The user receives a notification: “Your order has arrived!”
- Option to rate the order and tip the driver appears.

For you guys, please write your walkthrough yourselves!

It does have to follow this exact pattern, since ChatGPT has a habit of making everything as bullet points. You can use a mixture of paragraphs and steps to show your walkthrough.

Good luck, girls and boys!