General Instructions: Carefully read the following instructions before attempting the paper.

- Except your Roll No and Section, **DO NOT WRITE** anything on this paper.
- The Final Exam consists of 4 questions on 2 printed sides of 1 page.
- In case of any ambiguity, you may make assumptions, but your assumption must not contradict any statement in the question paper.
- **DON'T** share your program, if your code is matched to any member of your class, both will get **straight F** in the course without asking who shared or who magically copied.

Submission Instructions:

- You must comment your student ID on top of each file. (Line#1 of your code).
- Name the .c file for each question according to Roll_No e.g. **k22-xxxx_Q1.c, k22- xxxx _Q2.c** etc.
- Create a ZIP folder of all your solutions and copy it in the local storage with the title **K22-xxxx_A.**
- Submission are on local storage that can be accessed using win+r keys and entering \\172.16.5.41 address in the dialog box.
- Enter your username as **khifast\K22xxxx** and its assigned password.
- Zip folder needs to be pasted in the **"Exam Submission\teacherName"** folder

Time: 160 Minutes **Max Points:**100 Points

Task 1: **[30 mins, 25 points]**

Create a base class called Course that contains common properties and methods for all courses. The class has attributes such as name, course_code, credithours, and instructor. You define methods such as print_details() which will be override in the derived class.

Next, you create several specific course classes that inherit from the Course class. For example, you create a ThoeryCourse class that has additional attributes such as projects and mid1 and mid2 and final marks, and a LabCourse class that has attributes such as lab_tasks and lab_mid and lab_final marks. Both classes have get_grade() function which generates grades based on their evaluation criteria.
Then, you create a generic function called display_grade() that takes any Course object either TheoryCourse or LabCourse as an argument and calls the get_grade() function.

Define a generic filter_courses() function to filter courses by field value. It takes an array of courses, a second parameter indicating the field to filter by (e.g., "instructor" or credit hours). The function should call print detail functions for only those courses that match the specified value.

Task 2: **[30 mins, 25 points]**

Write a C++ Program that defines three classes: Passenger, Taxi, and two subclasses of Taxi, Uber and Careem.

1. Passenger has three private data members: name, age, and destination. It has one constructor which takes in name, age, and destination as arguments, and three member functions: getName(), getAge(), and getDestination(), which return the corresponding data members.

2. Taxi is an abstract class with five data members: make, model, year, passengerCount, and passengers, which is an array of pointers to Passenger objects. make, model, and year are initialized by its constructor, and passengerCount is initialized to 0. It has five member functions: getMake(), getModel(), getYear(),

3. getPassengerCount(), and printPassengers(). The last one prints the name and destination of all passengers in the taxi. Two of its member functions are pure virtual functions: addPassenger() and calculateFare().

4. Uber and Careem are subclasses of Taxi. They inherit all data members and member functions from Taxi and implement their own versions of the addPassenger() and calculateFare() functions.

5. addPassenger(Passenger* p) function in Uber adds a passenger to the passengers array if there is room for it (less than 4 passengers). addPassenger() function in Careem adds a passenger to the passengers array if there is room for it (less than 3 passengers).
6. calculateFare(double distance, bool isPeakHour) function in Uber calculates the fare based on a base fare of 50, a distance rate of 5.5, and a peak hour rate of 1.2 if isPeakHour is true.

The main() function creates three Passenger objects, two Taxi objects (Uber and Careem), and adds passengers to each taxi. It then calculates and prints the fares for each taxi and prints the passengers in each taxi.

Task 3: **[30 mins, 25 points]**

You have been tasked with developing a platform for Spotify, a music shop that allows customers to book or rent the songs for the specific period of time. The system must allow customers to select up to five songs offerings, with an additional backup song in case one of their chosen songs becomes already rented or booked. Each songs offering can have a maximum of forty copies, and any songs offering with less than ten rent booking will be canceled.
In addition to renting for songs, the system must also calculate the rent fees for the songs that a customers has rented for and send this data to the accounting system. Finally, singers must be able to access the system to view the songs they will be offering and see which customers have rented for their songs.
Create a class called "Songs" with the following attributes Songs_code (string) ,song_name (string), max_copies (integer), current_renters(integer), singer(string) fee(float) and methods:

1. Constructor method to initialize the attributes ✓
2. Getter and setter methods for each attribute ✓
3. Method to rent a customer for the songs (checks if the maximum number of customers has been reached and updates the current number of customers)
4. Method to drop a customer from the song (updates the current number of customers)
5. Friend function to calculate the total fee for a customer rented songs (sums up the fees for each song)
6. Overloading of the << operator to display the song details.

Create a class called "Customer" with the following attributes customer_id (string), name (string), rented_songs (Songs list), backup_songs (Songs list) and methods

1. Constructor method to initialize the attributes
2. Getter and setter methods for each attribute
3. Method to rented for a songs (checks if the songs is already full and adds the customer to the songs)
4. Method to drop a songs (removes the customer from the song)
5. Method to add a backup song(checks if the backup song is already fully booked)
6. Friend function to calculate the total fee for the customer rented song (calls the friend function in the Song class to calculate the rental fees)
7. Overloading of the << operator to display the customer details.

Finally, you create a main function that simulates the rental process. You create several Songs objects and Customer objects and use the methods in the Songs and Customer classes to simulate the rental process. You also use the friend function and operator overloading to calculate rental fees and display information.

Task 4: **[30 mins, 25 points]**

Implement a system for DevDay, they need to keep track of their participants. Participants' attributes include their name, roll_num, and competition name in which they are participating. The program must be able to add new participants, update participant details, and generate a report of the current enrolled participants.
Design a scenario for your program that involves file handling in C++. The scenario should include the following:

1. A menu system that allows the user to choose from options such as adding a new participant, updating an existing participant, generating a report for all participants competition-wise, and exiting the program.
2. The program should read the existing participant record from a text file and store the data in a data structure such as an array.
3. When the user adds a new participant, the program should append the data to the text file.
4. When the user updates an existing participant, the program should modify the data in the text file.
5. When the user generates a report, the program should read the data from the text file and output it to the console. The information should include the count of participants each competition-wise.