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

Assignment # 1

Subject: Object Oriented Programming - CS1004	**Post Date: 06/02/2025**
Total Marks: 50	**Due Date: 25/02/2025, 11:59 PM**
Course Instructors: Ms, Bakhtawar, Ms. Sobia Iftikhar, Ms. Abeer Gauher, Ms. Nida, Mr. Basit Ali, Ms. Atiya Johkio, Ms. Sumaiyah Zahid, Mr. Minhal Raza, Ms. Abeeha Sattar, Ms. Rafia	

Instructions to be strictly followed.

- Each student should submit these files:
  - ***A zip of all source files named as "A1-Q#[StudentID]" where # is the question number and Student ID is your ID.***
  - ***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 "A1-[StudentID].docx".***
  - ***All the submissions will be made on Google Classroom.***
- Each output should have STUDENT ID and NAME of the student at the top.
- It should be clear that your assignment would not get any credit if the assignment is submitted after the due date.
- Zero grade for plagiarism (copy/ cheating) and late submissions.

Question 01. **[Marks 10]**

FAST has many sports enthusiasts, and to encourage their growth, the university wants to implement a mentorship program. In this program, mentors guide learners (students), but each mentor has a limited capacity to manage a fixed number of learners. The university sports mentorship system involves two main roles: mentors and learners (students).

The Student class has data members such as studentID, name, age, sportsInterests[], and mentorAssigned. A student can register for mentorship under a mentor by calling the registerForMentorship(Mentor m) method. The student can also view the mentor's details and update their sports interests through viewMentorDetails() and updateSportsInterest(Sport s) methods, respectively.

The Mentor class, on the other hand, has data members like mentorID, name, sportsExpertise[], maxLearners, and assignedLearners[]. The maxLearners attribute specifies the maximum number of learners the mentor can handle. The mentor can assign learners using the assignLearner(Student s) method, as long as they have available capacity. If a learner needs to be

removed, the removeLearner(Student s) method can be used. The mentor can also view their list of assigned learners by calling viewLearners() and provide guidance using the provideGuidance() method.

The Sport class defines the sports offered for mentorship. It includes attributes like sportID, name, description, and requiredSkills[]. The mentor can add new skills required for a specific sport using the addSkill(Skill s) method, and they can also remove skills from the list using removeSkill(Skill s).

The Skill class defines the various skills related to each sport, including data members like skillID, skillName, and description. This class provides methods such as showSkillDetails() to display the skill details and updateSkillDescription(String newDescription) to update a skill's description. In this system, each mentor has a limited capacity to handle learners. For example, a mentor named Ali who specializes in Tennis can only mentor up to three students at a time. When a student like Saad, who is interested in Tennis, registers for mentorship, Ali can assign him as a mentee if there is space. If Ali already has three learners, Saad cannot be assigned until a slot becomes available. If Saad later decides to stop the mentorship, Ali can remove them, freeing up space for a new student. The system allows mentors to guide students on their sports skills, providing training and advice to improve their performance.

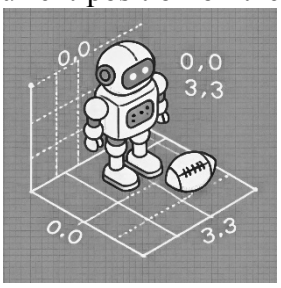
Question 02. [Marks 10]

Football Game Simulation

Your task is to create a football game simulation where two teams, each with a robot player, attempt to move a ball towards a predefined goal. The game determines the winner based on which team reaches the goal in the fewest number of hits.

The Robot class represents a robot and should have two key attributes: name, which holds the name of the robot as a string, and hits, which keeps track of the number of hits made by the robot. The class must include a method hitBall(int &ballX, int &ballY, const string &direction). This method will update the ball's position based on the direction given by the robot (either up, down, left, or right), and it will also increment the robot's hit count each time the ball is hit.

For ball position on the field specify by x and y, representing the ball's current position on the field. The class should provide methods such as getX() and getY() to return the current x and y coordinates of the ball, respectively. Additionally, the method move(int dx, int dy) will update the ball's position based on the movement made by the robot, and the method getPosition() will return the ball's position as a tuple (x, y) formate..



Goal's position on the field designed by x and y, which define the coordinates of the goal (set to (3, 3) for this assignment. isGoalReached(int ballX, int ballY), which will check whether the ball's position matches the goal's coordinates.

The Team class represents a football team, which consists of robot player. The team has a teamName attribute to store the name of the team and a robot attribute that holds a pointer to a Robot object, which represents the player.

The Game class manages the entire game flow. It holds two Team objects (teamOne and teamTwo), a Ball object, and a Goal object. The class includes the method startGame(), which starts the game, letting each team play its turn. The play(Team *team) method simulates the game for a given team, moving the ball towards the goal and counting the number of hits made by the robot. After both teams have played, the declareWinner() method compares the number of hits made by each robot and declares the team with the fewer hits as the winner.

you are required to design and implement a C++ program that simulates a simple football game. The program should follow Object-Oriented Programming (OOP) principles, particularly focusing on "Has-A" relationship

Question 03. [Marks 10]

Imagine you are developing a vehicle rental application that allows users to register, update their details, and rent vehicles based on certain eligibility criteria. The rental system must consider different types of users and vehicles. Users will provide their personal information, including details such as their age, contact number, and license type, while the system will offer a range of vehicles with varying rental conditions.

The application needs to handle user registration, allowing them to update their details as needed, and rent vehicles by checking eligibility based on their license type. Once a user is registered, they can view a list of available vehicles and choose one to rent. The system should check the user's eligibility to rent a specific vehicle based on their license type, and it should ensure users can only rent vehicles they are qualified for. You are asked to create a vehicle rental system where users can register with their details, update their personal information, and rent vehicles. The vehicles will have different types and rental conditions, and users will be restricted from renting vehicles they are not qualified for. The system will manage the following:

1. Users can register and provide information such as:
    - Age
    - License type (e.g., Learner, Intermediate, Full)
    - Contact information
    - User ID

2. Once registered, users can update their details, including their name, age, and license type, which will be stored in the system.

3. The system will display a list of available vehicles, with attributes such as:
    - Model
    - Rental price per day
    - Eligibility for different license types (e.g., Learners can only rent certain vehicles)

4. The system will check if the user meets the requirements to rent a specific vehicle based on their license type.

5. If the user is eligible for a vehicle, they can proceed with the rental; otherwise, they will be informed that they cannot rent that particular vehicle.

To store the available vehicles, a pointer array will be used. This allows for dynamic management of vehicle objects, providing flexibility for adding or removing vehicles as needed without resizing fixed arrays. The vehicle objects should be created dynamically using pointers, and the array will store the addresses of these objects. The system will access and modify vehicle details using these pointers.

By using a pointer array, the system ensures efficient memory usage and the ability to scale the number of vehicles as needed.

- Allow the user to register with their information, including personal details and license type.
- Implement functionality for users to update their information at any time.
- Create a dynamic array of pointers for vehicles. Each pointer in the array will point to a dynamically allocated vehicle object.
- For each vehicle, specify eligibility requirements based on license type (e.g., Learners, Full license).
- Based on the user's license type, check if they are eligible to rent a vehicle.
- Ensure that users with different license types (e.g., Learner, Intermediate, Full) have access to different vehicle options.
- If eligible, allow users to rent a vehicle, displaying the rental details and the vehicle they have selected.
- If not eligible, inform the user that they cannot rent that vehicle

Question 04. [Marks 10]

FAST's transportation system currently has a troublesome process for student registration, semester payments, and to verify it manually on points. They want you to automate the entire process to streamline the system.

The system should

- Allow students to register for the transportation service.
- Enable students to pay semester fees to keep their transportation cards active.
- Manage routes, including pick-up and drop-off stops for students.
- Record attendance automatically when students tap their cards on the bus.
- Manage bus routes with stops and assign students to appropriate stops.

Identify the required classes, their data members, and member functions. Then, create a class diagram to represent the system. Based on your class diagram, write a C++ program to implement the system.

You must follow all the OOP rules learned up to the 5th week. This means using proper getters and setters, constructors, destructors, and making use of constant and static variables where needed.