National University of Computer and Emerging Sciences

Object Oriented Programming (CS 1004)

Date: Feb 28th 2024

Course Instructor(s)

Mr. Basit, Mr. Minhal, Ms.Sumaiya, Ms.Sobia, Ms.Abeeha, Ms.Bakhtawer, Ms.Abeer, Ms.Atiya, Ms.Rafia

Sessional-I Exam

Total Time: 1 Hour
Total Marks: 30
Total Questions: 03

Semester: SP-2024
Campus: Karachi
Dept: School of Computing

Student Name | Roll No | Section | Student Signature

CLO #1: Discuss knowledge of underlying concepts of object-oriented paradigm like abstraction, encapsulation, polymorphism, inheritance etc **[20 minutes, 10 marks (6*1 + 2*2)]**

Q1: Write short answers (2-3 lines) for the following questions:

a. A good programming practice is that the arguments of setter functions are constant, and getter functions themselves are constant functions. Why is that so? Please relate your answer to a real-world scenario.

b. In object-oriented programming, why is it considered good practice to encapsulate data within classes and provide access to it through methods?

c. In object-oriented programming, methods of a class are generally made public so that they can be called from anywhere. However, can any method not be private? How can functions be made private? Please justify your answer with a real-world use case.

d. Why do we need a user-defined copy constructor if the compiler-defined copy constructor is already available?

e. What is the need to keep a static attribute within a class when it could be declared as a global variable?

f. Which non-constant member function can be called by a constant object? Explain your answer.

g. Find the Error and write the reasons in 2-3 lines.

```
class ConstantValueContainer {
private: const int value;   const int* const ptr;
public:
ConstantValueContainer(int   initialValue)   :
value(initialValue), ptr(&value) { }
void displayValue() const {
cout << "Value through the constant pointer: "
<< *ptr <<endl;  }
void modifyValue() { *ptr = 99; }
};
int main() {
ConstantValueContainer container(42);
container.modifyValue();
container.displayValue(); }
```

h. Find the Error and write the reasons in 2-3 lines.

```
class Complex
{
private:
double x,y;
static int z;
public:
Complex (double = 0.0);
static int doSomething() { z=2* y; return z;
};
};
```

***CLO #2: Identify real world problems in terms of objects rather than procedure.* [20 minutes, 10 marks (2+ 4*2)]**

Q2: Develop a Program that organize digital cricket match where two players, Player 1 and Player 2, are competing over 12 balls. The game, named "Cricket Showdown," involves each player taking turns to score runs on every ball. The twist is, they have to enter their scores, ranging from 0 to 6, and if a player mistakenly inputs a score outside this range, the score won't count, but the ball will still be marked. You should not violate any principle of Object-oriented programming.

1. Create a class "player" consisting of following members with appropriate datatypes:
   - ***ballScores[12]***: denotes an array wherein the scores corresponding to each ball are stored.
   - ***PlayerName***: representing the player name.
   - ***TotalScore***: containing count total score for each player.

2. Create a class "Game" consisting of following members with appropriate datatypes:
   - ***playGame(Player& player)***: each player gets their turn to face the 12 balls. The function prompts the player to enter the score for each ball.
   - ***ValidateScore()***: if the score entered is not between 0 and 6 (inclusive), it won't be considered, but the ball will still be recorded.
   - ***findWinner(Player& player1, Player& player2)***: the winner is determined based on the total scores.
   - ***displayMatchScoreboard(Player& player1, Player& player2)***: This function displays the detailed summary of the match. In summary, each player's name and their score against each ball should be in ascending manner with average score and total score for each player, giving a comprehensive overview of their performance.

CLO #4 Design and assess small and medium scale C++ programs using OOP principles.
[20 minutes, 10 marks (2+2+6)]

Q3: In the vast expanse of space, a Space Exploration Agency endeavors to push the boundaries of human knowledge and exploration. To achieve its ambitious goals, the agency relies on meticulously designed spacecraft, dedicated missions, and highly trained astronauts.

The space agency is also working on software that keeps track of its missions, and the spacecrafts and astronauts associated with those missions. Each Astronaut has a name, expertise, and a status (assigned to a mission or not). Each spacecraft has a unique name, capacity, and mission readiness status. Each mission has a unique name, duration, mission requirements, and destination. It also stores information about the spacecraft used for the mission, and the astronauts participating in the mission. One mission can have only one spacecraft, and can have a maximum of **n** astronauts, *where n is the capacity of the spacecraft participating in the mission*. It has the functionality to assign a spacecraft to the mission, which ensures that the spacecraft is mission ready before assignment, and then updates the maximum number of astronauts that can be assigned to the mission. It also has the functionality to add an astronaut to the mission, which first ensures that the astronaut is not assigned to any other mission, and then checks if their expertise matches at least one of the mission requirements. This function also validates that no more than **n** astronauts can be added to the mission. For the above scenario, your task is to identify the attributes and functions of all three entities, and write a C++ program for it. You need to match the following requirements:

1. Implement the astronaut class with a parameterized constructor, using member initialization list and also provide destructor.
2. Implement the spacecraft class with a parameterized constructor, using member initialization list.
3. Perform the following tasks:
   i. Implement the mission class with a parameterized constructor that accepts all class members as argument, except the astronauts. You need to dynamically allocate an array for astronauts based on the spacecraft size.
   ii. Implement the function for adding a spacecraft as defined in the scenario.
   iii. Implement the function for adding an astronaut as defined in the scenario.

---------- The End ----------