ಭಾರತೀಯ ಮಾಹಿತಿ ತಂತ್ರಜ್ಞಾನ ಸಂಸ್ಥೆ ರಾಯಚೂರು
भारतीय सूचना प्रौद्योगिकी संस्थान रायचूर
Indian Institute of Information Technology Raichur

LAB - 5

Introduction to OOP

Submission Guidelines

- Ensure your system is in 'No Network Icon' 🌐.
- Do NOT Hide Your Taskbar.
- Create a new folder named `LAB-5`.
- Inside `LAB-5`, create two question files named in the following format:
  `[RollNumber]_[LabName]_[QuestionNumber]`
  (e.g., `12345_Lab5_Q1.cpp` and `12345_Lab5_Q2.cpp`)

Lab Timing and Submission

- Lab Time: 10:30 AM - 12:30 PM
- Submission Deadline: Till 12:35 PM (Submit on Classroom)
- No Extensions: Late submissions will not be accepted.
- Failed to follow any of the submission guidelines will result in deduction of marks in that particular lab.
- Viva: 12:30 PM - 1:00 PM (Marks will be assigned based on viva performance)

Question 1:- (40 points)

C++ Assignment: RPG Game Inventory and Player Stats using Templates.

Objective: Design a C++ program using templates to manage RPG player stats and inventory, ensuring input for multiple players and displaying their details for a basic RPG game.

Implementation of string in C++ : To use the string data type in C++, you need to include the <string> header. This allows you to declare and use string variables such as string name;. Additionally, you must use the std namespace or prefix string with std::.

```
#include <iostream>
#include <string> // Required to use string type

using namespace std;

int main() {
  string itemName;
  cout << "Enter item name (no spaces): ";
  cin >> itemName;
  cout << "Item name is: " << itemName << endl;
  return 0;
}
```

Note: This program uses cin to take input for a string without spaces. To read strings with spaces, you would normally use getline(), but this assignment avoids that for simplicity.

Requirements:

1. **Item Structure**
    - Create a struct Item with:
        - int id; // Item ID
        - string name; // Name without spaces
2. **Template Class: InventoryBox**
    - Private members:
        - A fixed-size array to store 5 items
        - An integer to count how many items are added
    - Public methods:
        - void setCount() — resets the item count
        - void addItem(T item) — adds an item if inventory isn't full
        - void displayItems() — shows all items
3. **Template Class: BaseStats**
    - Protected member:
        - Array of 3 stats: health, strength, and defense
    - Public methods:
        - void setStats(T h, T s, T d) — sets the 3 base stats
        - void displayBaseStats() — shows the base stats
4. **Template Class: ExtendedStats** (Derived from BaseStats)
    - Adds 2 new stats: speed and agility
    - Methods:
        - void setExtendedStats(T h, T s, T d, T sp, T ag) — sets all 5 stats
        - double computeAverage() — returns average of 5 stats

        - void displayAllStats() — shows all stats and the average

5. **Main Function**
    - Asks user for number of players (max 10)
    - **For each player:**
        - Enter stats: health, strength, defense, speed, agility
        - Enter number of items (max 5)
        - Enter item ID and name (no spaces)
    - **After input:**
        - Display each player's stats and average
        - Display their inventory

Constraints:

- No dynamic memory (no new/delete)
- No use of getline() or cin.ignore()
- Max 10 players, 5 items each
- Item names should not contain spaces

Sample Input:

Enter number of players: 2

--- Player 1 ---
Enter health, strength, defense, speed, agility: 100 80 70 90 85
Enter number of items (max 5): 2
Enter item ID 1: 101
Enter item name (no spaces): Sword
Enter item ID 2: 102

Enter item name (no spaces): Potion

--- Player 2 ---
Enter health, strength, defense, speed, agility: 120 95 88 75 80
Enter number of items (max 5): 3
Enter item ID 1: 201
Enter item name (no spaces): Axe
Enter item ID 2: 202
Enter item name (no spaces): Shield
Enter item ID 3: 203
Enter item name (no spaces): Elixir

Sample Output:

=== **Player 1** ===
Health: 100, Strength: 80, Defense: 70
Speed: 90, Agility: 85
Average: 85
Inventory:
  ID: 101, Name: Sword
  ID: 102, Name: Potion

=== **Player 2** ===
Health: 120, Strength: 95, Defense: 88
Speed: 75, Agility: 80
Average: 91.6
Inventory:
  ID: 201, Name: Axe
  ID: 202, Name: Shield
  ID: 203, Name: Elixir

Question 2:- (60 points)

Rock-Paper-Scissors

You're simulating a competitive **Rock-Paper-Scissors (RPS)** tournament between two players. Each player has a **sequence of fighters**, each of which is either `Rock`, `Paper`, or `Scissors`, and each has a **base strength** (an integer).

In a round, fighter `i` from Team-A fights fighter `i` from Team-B.

- If a fighter **wins** by RPS rules, they deal `+10` bonus power.
- If a fighter **loses**, they deal `-5` penalty.
- If it's a draw, no bonus or penalty is applied.

The **actual power** is `base + bonus`. The team with the **higher total adjusted power** wins.

Object-Oriented & Template Requirements:

Abstract Base Class:

```
class Fighter {
public:
  virtual int basePower() const = 0;
  virtual std::string type() const = 0;  // "Rock", "Paper", "Scissors"
  virtual ~Fighter() = default;
};
```

Marker Types:

```
struct Rock {};

struct Paper {};

struct Scissors {};
```

Template Fighter Class:

```
template<typename Move>

class RPSFighter : public Fighter {

  int power;

public:

  RPSFighter(int p);

  int basePower() const override;

  std::string type() const override;

};
```

RPS Logic Function:

```
int matchOutcome(const Fighter& A, const Fighter& B);
```

- Returns **+10** if A wins
- Returns **–5** if A loses
- Returns **0** if draw

Winner Function:

```
int getWinner(

  const std::vector<std::unique_ptr<Fighter>>& teamA,

  const std::vector<std::unique_ptr<Fighter>>& teamB

);
```

- Returns `1` if A wins, `-1` if B wins, `0` if draw

Input:

Example:

- **Team A:**
  Rock, 20
  Scissors, 30
  Paper, 25

- **Team B:**
  Scissors, 25
  Rock, 30
  Rock, 20

Output:

```
1   // Team A wins
```

Matchups:

- Rock vs Scissors → A wins → 20 + 10 = 30 vs 25
- Scissors vs Rock → A loses → 30 - 5 = 25 vs 30
- Paper vs Rock → A wins → 25 + 10 = 35 vs 20

Total A = 30 + 25 + 35 = 90
Total B = 25 + 30 + 20 = 75

Team A wins → return 1

Constraints:

- 1 ≤ `teamA.size()` == `teamB.size()` ≤ 100
- Base power of each fighter: 1 ≤ power ≤ 1000