CS 200 - Introduction to Programming

Assignment 3

(Part-2)

Due Date: **25th November 2022**

- **Your code should not crash in any case.**
- **All data members should be private/protected.**
- **For each class write Setters and Getters for each data member.**
- **For each class write all types of Constructors and Destructors, i.e. Default constructor and Destructor, Parameter constructor, Copy constructor.**
- **For copy constructors(in case of dynamic memory) please declare the memory first, with the help of "new" keyword and then copy the content using loop.**
- **Plagiarism will lead to straight F in the course.**
- **Rename the files as <rollNumber>_exercise_1.c and <rollNumber>_exercise_2.c. Place the files in a zip folder, name it as <rollNumber>_PA3.2 and submit on LMS.**

Exercise-1

(60 Marks)

In the part-2, you will have to implement following ***Administrator*** operations of LUMUN Accommodation system:

- **emptyOccupancies()** **10 Marks**
  It generates a report (contains just the room number) of currently empty rooms of each accommodation facility (both male and female hostels)

- **reservedOccupancies()** **10 Marks**
  It generates report (contains all the booking information) of currently reserved rooms of each accommodation facility.

- **waitingListInfo()** **10 Marks**
  It generates report (contains all the booking information) of currently waiting Delegates of each accommodation facility.

- **generateIncome()** **20 Marks**
  It generates report of the income from the current bookings of each accommodation facility along with grand total

General Instructions:

You can either use the code provided for part-1 or you can continue with your own solution. In both cases, you are free to implement new classes and can also tweak the provided code.

Write a main function to test the program for both parts. The main function shall provide a two-level menu. First level is to choose between the Delegates and Administrative functions and the second level is the respective functions for the above actors.
Correctly displaying menus of each option. 10 Marks

First Level:

1. Room Reservation (Delegates Operations)
2. Administrative Operations
3. Quit

Exercise-2

(40 Marks)

In this exercise you will build a simple board game. A board game comes with a board, divided into 40 squares, **a pair of six-sided dice**, and can accommodate 2-6 players. It works as follows:

1. All players start from the first square.
2. One at a time, players take a turn, roll the dice and advance their respective tokens on the board.
3. A round consists of all players taking their turns once.
4. The winner will be the player that first advances beyond the 40th square.

To do:

- First implement the prototype of the board game using the following classes:
- Class **Game**: encapsulates the logic of the game (start state, the structure of a round, ending conditions). **(21 Marks)**
- Class **Die**: provides random numbers in the required range. [Hint: to get the numbers in a range use following code, v1 = rand() % range;] **(7 Marks)**
- Class **Player**: stores the state of each player in the game and performs a turn Once You have completed the basic functionality, you will extend the implementation of board game to play with money: **(12 Marks)**
    - Players have money. Each player starts with 7,000.
    - The amount of money changes when a player lands on a special square:
    - {Squares 5, 15, 25, 35 are bad investment squares: a player has to pay 5,000. If the player cannot afford it, he gives away all his money.
    - {Squares 10, 20, 30, 40 are lottery win squares: a player gets 10,000.
    - The winner is the player with the most money after the first player advances beyond the 40th square. Ties (multiple winners) are possible.

Extend the below skeleton code to complete the game!

```
#include<iostream>
using namespace std;
class Die{
protected:
  int number;
//define the other requirements(setter,getter, constructors etc)
// No other functions are allowed
};
class Player{
protected:
  int *scores;
  int size;
//define the constructors, setter getters and destructor
//No other helper functions are allowed
};
class Game :public Die, public Player{
 private:
  int round;
  int money_size;
  int *money;
 public:
      void playGame(); //write your game logic here.
      void print();  //print money and scores of all players
      void CheckWinner(); //print the winner player info(money, score,
current round number) if there is any winner(s)
      //define other requirements.No other helper functions are allowed
};
int main(){

      int *arr1; //players scores array
      int size;
      cout << "Enter the number of players :  ";
      cin >> size;
      while (size < 2 || size>6){
            cout << "Enter valid number of players : ";
            cin >> size;
      }
      arr1 = new int[size];
      for (int i = 0; i < size; i++){
            arr1[i] = 0;
      }
      int *arr2 = new int[size]; //array used to store money
      for (int i = 0; i < size; i++)
      {
            arr2[i] = 7000;
```

```
    }
    Game g1(1,size, arr2, 6, arr1, size); //starting from round#1
    cout << "Before Starting game : " << endl;
    g1.print();
    cout << endl << "After Starting game " << endl;
    g1.playGame();
    g1.playGame();
    g1.playGame();
    g1.CheckWinner();
    g1.playGame();
    g1.CheckWinner();
    g1.playGame();
    g1.CheckWinner();
    return 0;
}
```

Good luck and Happy Coding! 😇