Trial Mid Term Practical

1. A shepherd has 6 sheep. Every morning he counts them to make sure that no uninvited guest visited during the night. Develop a console application that counts the number of sheep present, absent and presence unknown. User should input true or false for every sheep. Extract the logic for the counting in a separate method that counts the number of sheep present and absent in the array (true means present, false means absent, anything else means presence unknown).

Hint: Don't forget to check for bad values like null/undefined. If you identify such value output 'presence unknow' in the console. Try to implement exception handling where applicable for extra points.

Example Input and Output:

Output: Start inserting sheep presence:

Input: true, true, none, false, true, false

Output: 3 sheep are present

2 sheep are absent

1 sheep presence unknown

using System;

class Program  
{  
    static void Main()  
    {  
        Console.WriteLine("Start inserting sheep presence (true/false/any other value for unknown):");  
        string[] inputs = Console.ReadLine().Split(',');  
          
        // Convert input strings to nullable booleans  
        bool?[] sheepPresence = new bool?[inputs.Length];  
        for (int i = 0; i < inputs.Length; i++)  
        {  
            sheepPresence[i] = ParseSheepPresence(inputs[i].Trim());  
        }

        // Count sheep  
        CountSheep(sheepPresence);  
    }

    static bool? ParseSheepPresence(string input)  
    {  
        if (bool.TryParse(input, out bool result))  
        {  
            return result;  
        }  
        return null; // Presence unknown  
    }

    static void CountSheep(bool?[] sheep)  
    {  
        int present = 0, absent = 0, unknown = 0;

        foreach (var s in sheep)  
        {  
            if (s == true)  
                present++;  
            else if (s == false)  
                absent++;  
            else  
                unknown++;  
        }

        Console.WriteLine($"{present} sheep are present");  
        Console.WriteLine($"{absent} sheep are absent");  
        Console.WriteLine($"{unknown} sheep presence unknown");  
    }  
}

1. There are 8 students in one group in a music school. The maximum number of students that can be part of one group is 8 and no more places can be allocated for any extra students. Students in one group can be of different ages. Develop a console application which finds the youngest and the oldest student in the class and outputs their age in the console. If there are more students of the same age, together with the age output a message saying how many students of that age there are in the group. If there is only one youngest or oldest students, do not output the number of students at that age, but only output the age.

Hint: For cleaner code, you can implement separate methods for finding the max and min age and number of students at that age. You should not allocate more memory than total number of students that could fit in one group. Try to implement exception handling for extra points.

Example Input and Output:

Output: Start inserting students age:

Input: 10, 8, 9, 12, 15, 7, 7, 8

Output: Youngest student in the group is 7 years old. There are 2 students at that age in the group.

Oldest student in the group is 15 years old.

using System;  
using System.Linq;

class Program  
{  
    static void Main()  
    {  
        Console.WriteLine("Start inserting students' ages (comma-separated, max 8 students):");  
          
        try  
        {  
            string[] inputs = Console.ReadLine().Split(',');  
            if (inputs.Length > 8)  
            {  
                Console.WriteLine("Error: Maximum number of students in a group is 8.");  
                return;  
            }

            int[] ages = inputs.Select(input => int.Parse(input.Trim())).ToArray();  
            FindYoungestAndOldest(ages);  
        }  
        catch (FormatException)  
        {  
            Console.WriteLine("Invalid input. Please enter only numbers.");  
        }  
        catch (Exception ex)  
        {  
            Console.WriteLine($"An unexpected error occurred: {ex.Message}");  
        }  
    }

    static void FindYoungestAndOldest(int[] ages)  
    {  
        int minAge = ages.Min();  
        int maxAge = ages.Max();

        int minCount = ages.Count(age => age == minAge);  
        int maxCount = ages.Count(age => age == maxAge);

        Console.Write($"Youngest student in the group is {minAge} years old. ");  
        if (minCount > 1) Console.WriteLine($"There are {minCount} students at that age.");  
        else Console.WriteLine();  
          
        Console.Write($"Oldest student in the group is {maxAge} years old. ");  
        if (maxCount > 1) Console.WriteLine($"There are {maxCount} students at that age.");  
        else Console.WriteLine();  
    }  
}

1. Develop a console application which will output student with their respected details and GPA.

Define a base class Student, which will store following information:

o Name (text)

o Surname (text)

o ID (whole number)

o Year of Study (whole number).

o List of All Grades (whole numbers)

Name, Surname, ID and Year of Study should be private. List of All Grades should be public.

For all above properties implement encapsulation by making use of get and set properties.

For the Student class implement a default constructor.

For the Student class implement a parameterized constructor.

For the Student class implement a destructor.

In the Student class implement a method for printing information for the student (Name, Surname, ID, Year of Study) named PrintInfo.

In the student class define a method for calculating and printing the GPA (Grade Point Average) per student called CalcucalteGPA (accepting a list of all grades), but do not implement it. The GPA should be a decimal number. Grades should be whole numbers.

Define a class GPA which is inheriting from the Student class.

In the GPA class inherit the parameterized constructor from the Student class.

In the GPA class override the implementation of the PrintInfo method from the Student class to print the Student grades this time.

In the GPA class implement the method CalcucalteGPA to calculate and print the GPA with 2 decimal places.

Define a class Course., which will store following information:

o List of All Courses (texts)

List of All Courses should be private.

In the Course implement a method PrintCourses which will print courses that student has been enrolled in.

For the Course class create a corresponding Interface and implement it accordingly.

Implement the main method:

o Ask the user to insert the student Name, Surname, ID, Year of Study and Grades.

o User should be inserting grades until “.” is detected. Once the user types “.”, no more grades should be inserted.

o Inserted grades must be between 5 and 10. If the user inserts grade value that is less than 5 or greater than 10, a message: “You have inserted incorrect grade. Allowed grades should be between 5 and 10!” should be printed in the console and the incorrect grade should not be added to the List of grades. Grades reading should proceed with next grades until “.” input has been detected.

o Based on the values inserted calculate the student’s GPA and output Name, Surname, ID, Year of Study (using get and set methods already implemented). At the end also output the GPA.

Hint: Make sure that you check whether all entered grades are between 5 and 10. Try to implement exception handling where applicable for extra points.

Expected Input and Output:

Output: Please insert Student name:

Input: John

Input: Smith

Output: Please insert Student surname:

Output: Please insert Student ID:

Input: 1234

Output: Please insert Year of study:

Input: 2021

Output: Please start inserting student grades between 5 and 10. To stop, please insert ".":

Input: 5

6

7

9

10

Output: Name: John

Surname: Smith

ID: 1234

Year of Study: 2021

GPA: 3.80

using System;  
using System.Collections.Generic;  
using System.Linq;

interface ICourse  
{  
    void PrintCourses();  
}

class Course : ICourse  
{  
    private List<string> Courses = new List<string>();

    public void AddCourse(string course)  
    {  
        Courses.Add(course);  
    }

    public void PrintCourses()  
    {  
        Console.WriteLine("Enrolled Courses: " + string.Join(", ", Courses));  
    }  
}

class Student  
{  
    private string Name;  
    private string Surname;  
    private int ID;  
    private int YearOfStudy;  
    public List<int> Grades { get; set; } = new List<int>();

    public Student() { }

    public Student(string name, string surname, int id, int yearOfStudy)  
    {  
        Name = name;  
        Surname = surname;  
        ID = id;  
        YearOfStudy = yearOfStudy;  
    }

    ~Student()  
    {  
        Console.WriteLine("Student object destroyed.");  
    }

    public string GetName() => Name;  
    public string GetSurname() => Surname;  
    public int GetID() => ID;  
    public int GetYearOfStudy() => YearOfStudy;

    public void PrintInfo()  
    {  
        Console.WriteLine($"Name: {Name}\nSurname: {Surname}\nID: {ID}\nYear of Study: {YearOfStudy}");  
    }

    public virtual void CalculateGPA() { }  
}

class GPA : Student  
{  
    public GPA(string name, string surname, int id, int yearOfStudy) : base(name, surname, id, yearOfStudy) { }

    public override void CalculateGPA()  
    {  
        if (Grades.Count == 0)  
        {  
            Console.WriteLine("No grades available to calculate GPA.");  
            return;  
        }

        double gpa = Grades.Average();  
        double normalizedGPA = (gpa - 5) / 5 \* 4; // Normalizing to 4.0 scale  
        Console.WriteLine($"GPA: {normalizedGPA:F2}");  
    }  
}

class Program  
{  
    static void Main()  
    {  
        try  
        {  
            Console.Write("Please insert Student name: ");  
            string name = Console.ReadLine();

            Console.Write("Please insert Student surname: ");  
            string surname = Console.ReadLine();

            Console.Write("Please insert Student ID: ");  
            int id = int.Parse(Console.ReadLine());

            Console.Write("Please insert Year of Study: ");  
            int year = int.Parse(Console.ReadLine());

            GPA student = new GPA(name, surname, id, year);  
            Console.WriteLine("Please start inserting student grades between 5 and 10. To stop, please insert \".\":");

            while (true)  
            {  
                string input = Console.ReadLine();  
                if (input == ".")  
                    break;

                if (int.TryParse(input, out int grade))  
                {  
                    if (grade >= 5 && grade <= 10)  
                        student.Grades.Add(grade);  
                    else  
                        Console.WriteLine("You have inserted an incorrect grade. Allowed grades should be between 5 and 10!");  
                }  
                else  
                {  
                    Console.WriteLine("Invalid input. Please enter a number between 5 and 10.");  
                }  
            }

            student.PrintInfo();  
            student.CalculateGPA();  
        }  
        catch (FormatException)  
        {  
            Console.WriteLine("Invalid input detected. Please enter valid numeric values.");  
        }  
        catch (Exception ex)  
        {  
            Console.WriteLine($"An unexpected error occurred: {ex.Message}");  
        }  
    }  
}