# M1 Practice Questions

## Question 1

### Scenario:

Sam and John's instructor received a project from Machine Masters to create an application that computes the salary of employees depending on their employment type, whether they are permanent or temporary. The application will receive the basic salary as input, and the output will be the net salary. In order to enhance their development skills, the instructor tasked Sam and John with implementing the project.

Help them to complete the task using your C# coding skills.

### Functionalities:

In **abstract** class **Employee,** implement the below-given properties and methods.

| **Data Type** | **Property Name** |
|---|---|
| string | Id |
| string | Name |
| double | Salary |

| **Method** | **Description** |
|---|---|
| public bool ValidateEmployeeId() | This method is used to validate the employee id.<br>1. The employee id should have **five** characters.<br>2. The first two characters should be **"EM"**<br>3. The remaining three characters should be **numbers**.<br>If the above conditions are satisfied, then return **true**. Otherwise, return **false**. |
| public abstract Employee SalaryCalculation() | This method is used to calculate the salary of the employee, store the final value in the property **Salary** and return the Employee object.<br>This is an **abstract** method to **override** and use for calculation based on the employment type. |

In class **Permanent,** implement the below-given property and also inherit the class **Employee.**

| **Data Type** | **Property Name** |
|---|---|
| double | BasicPay |

In class **Temporary,** implement the below-given properties and also inherit the class **Employee.**

| **Data Type** | **Property Name** |
|---|---|
| int | HrsWorked |
| int | HrlyWages |

**Formula :**

1. Permanent employee salary = Basic Pay + DA + HRA

DA = 50% of Basic Pay
HRA = 30% of Basic Pay

2. Temporary employee salary = HrsWorked * HrlyWages

In **Program** class - **Main** method,

**1.** Get the values from the **user** as per the Sample Input.

**2.** Call the **ValidateEmployeeId** method. If this method returns true, then move on to step 3. Otherwise, display **"Invalid id".**

**3.** Call the **SalaryCalculation** method as per the employment type and display the result as per the Sample Output.

**Note:**

- Keep the properties, methods and classes as **public.**
- Please read the method rules **clearly**.
- Do not use **Environment.Exit()** to terminate the program.
- Do not change the given code template.

**Summary :**

Through working on this project, they have learned about **abstract** classes and methods in C#.

An **abstract class** is a class that cannot be instantiated and it may contain abstract and non-abstract methods.

An **abstract method** is a method without implementation, and it must be implemented by the derived classes.

**Sample Input 1:**

1.Permanent

2.Temporary

Choose the employee type

**1**

Enter the employee id

**EM123**

Enter the name

**Sam**

Enter basic pay

**8000**

**Sample Output 1:**

Employee id : EM123,  Name : Sam,  Salary : 1440

**Sample Input 2:**

1.Permanent

2.Temporary

Choose the employee type

**2**

Enter the employee id

**EM102**

Enter the name

**John**

Enter hours worked

**8**

Enter hourly wages

**700**

**Sample Output 2:**

Employee id : EM102,  Name : John,  Salary : 5600

**Sample Input 3:**

1.Permanent

2.Temporary

Choose the employee type

**2**

Enter the employee id

**A1102**

**Sample Output 3:**

Invalid id

## Question 2

The hospital management system allows staff to manage patient records online. Staff can log in using their credentials, add new patient records, update patient details, and check patient records. The system validates inputs to ensure data integrity and correctness. Exception handling is used to manage errors effectively.

**Component Specifications: PatientManager (Utility class)**

| Type (Class) | Method | Parameters | Responsibilities |
|---|---|---|---|
| PatientManager | validateCredentials | String username, String password | Validates the staff's username and password. Throws InvalidCredentialsException if invalid. |
| PatientManager | addPatient | String name, int age, String gender, String diagnosis | Adds a new patient record to the system. |
| PatientManager | updatePatientDetails | String name, int newAge, String newGender, String newDiagnosis | Updates the details of an existing patient in the system. Throws PatientNotFoundException if patient does not exist. |
| PatientManager | checkPatientRecord | String name | Returns the details of a given patient. Throws PatientNotFoundException if patient does not exist. |

**UserInterface Class (Main method)**

The main method in the UserInterface class prompts the user to enter their username and password to log in. If the credentials are valid, it allows the user to choose from options to add a new patient, update patient details, or check patient records. Each action calls the appropriate method from PatientManager to perform operations on the records. Exceptions are handled to provide meaningful error messages to the user.

**Additional Validation Conditions**

1. **Username and Password Validation:**
    - Username must be at least 6 characters long and can only contain alphanumeric characters and underscores.
    - Password must be at least 8 characters long and should include at least one uppercase letter, one lowercase letter, one digit, and one special character.
2. **Patient Details Validation:**

- Age should be a positive integer.
- Gender should be either "Male", "Female", or "Other".

**Explanation:**

In this scenario, the PatientManager class handles operations related to patient records management such as validating credentials, adding new patients, updating patient details, and checking patient records. Exceptions (InvalidCredentialsException and PatientNotFoundException) are used to handle errors gracefully and provide informative messages to the user via the UserInterface class.

**Sample Input/Output**

**Sample Input 1:**

yamlCopy codeEnter Username:staff1Enter Password:StaffPass123!Choose an Option:1. Add Patient2. Update Patient Details3. Check Patient Record4. ExitEnter Option:1Enter Patient Name:John DoeEnter Age:30Enter Gender:MaleEnter Diagnosis:FluPatient record added successfully.Enter Option:2Enter Patient Name to Update:John DoeEnter New Age:31Enter New Gender:MaleEnter New Diagnosis:Common ColdPatient details updated successfully.Enter Option:3Enter Patient Name:John DoePatient Details:Name: John DoeAge: 31Gender: MaleDiagnosis: Common ColdEnter Option:4Logout Successful.

**Sample Input 2:**

mathematicaCopy codeEnter Username:staff123Enter Password:Staff456!Choose an Option:1. Add Patient2. Update Patient Details3. Check Patient Record4. ExitEnter Option:1Enter Patient Name:Jane DoeEnter Age:-25Invalid age. Please enter a positive integer.Enter Age:25Enter Gender:FemaleEnter Diagnosis:MigrainePatient record added successfully.Enter Option:3Enter Patient Name:Jane SmithPatient not found.Enter Option:4Logout Successful.

## Question 3

A travel agency needs a new software application to calculate the total cost of travel packages for different types of travelers. The agency offers different types of travel packages, such as Adventure and Luxury packages, each with its own pricing and discount rules. After discussing the requirements with the travel agency, it was decided that using an interface would be the best approach to develop an application that accurately calculates the total cost based on the traveler type. The task of implementing the application is assigned to Sam and John.

Help them complete the task using your C# coding skills.

**Functionalities:**

Create an interface ITravelPackage, and implement the below-given properties and declare the method.

| Data Type | Property Name |
|---|---|
| string | PackageName |
| double | BasePrice |
| int | NumberOfDays |

| Method | Description |
|---|---|
| double CalculateTotalCost() | This method is used to calculate the total cost based on the travel package type and return it. |

The class AdventurePackage needs to implement the interface ITravelPackage.

| Method | Description |
|---|---|
| public double CalculateTotalCost() | This method calculates the total cost of the Adventure package and returns it. If the number of days is more than 5, a 10% discount is applied to the total cost. Calculate the total cost using the below-given formula. |

The class LuxuryPackage needs to implement the interface ITravelPackage.

| Method | Description |
|---|---|
| public double CalculateTotalCost() | This method calculates the total cost of the Luxury package and returns it. If the number of days is more than 3, a 15% discount is applied to the total cost. Calculate the total cost using the below-given formula. |

Formula: TotalCost = BasePrice * NumberOfDays

In the Program class - Main method,

1. Get the values from the user as per the Sample Input.
2. Call the methods accordingly and display the result as per the Sample Output.

**Note:**

- Keep the properties, methods, and classes as public.
- Please read the method rules clearly.
- Do not use Environment.Exit() to terminate the program.
- Do not change the given code template.

## Question 4

The trainer tasked with developing a new application for a bank that would be able to calculate interest rates for various account types, including savings and checking accounts. After thoroughly discussing the project requirements with the bank, he decided that utilizing the interface concept would be the best approach to developing an application that could accurately calculate interest rates based on each account type. He then delegated the implementation of the application to Sam and John.

Help them to complete the task using your C# coding skills.

**Functionalities:**

Create an interface **IAccount,** and implement the below-given properties and declare the method.

| Data Type | Property Name |
|---|---|
| string | AccountNumber |
| double | Balance |
| double | InterestRate |

| Method | Description |
|---|---|
| double CalculateInterest() | This method is used to calculate the interest based on the account type and return it. |

The class **SavingsAccount** needs to implement the interface **IAccount**.

| Method | Description |
|---|---|
| public double CalculateInterest() | This method is used to calculate the interest of the **Savings Account** and return it.<br><br>If the balance is less than 1000, then deduct **0.1%** of the **balance** as a penalty from the interest earned.<br><br>Calculate the interest based on the below-given formula. |

The class **CheckingAccount** needs to implement the interface **IAccount**.

| **Method** | **Description** |
|---|---|
| public double CalculateInterest() | This method is used to calculate the interest of the **Checking Account** and return it.<br><br>If the balance is less than 5000, then deduct **0.5%** of the **balance** as a penalty from the interest earned.<br><br>Calculate the interest based on the below-given formula. |

**Formula :**
**Interest** = Balance * InterestRate / 100

In **Program** class - **Main** method,

1. Get the values from the **user** as per the Sample Input.
2. Call the methods accordingly and display the result as per the Sample Output.

**Note:**

- Keep the properties, methods and classes as **public.**
- Please read the method rules **clearly**.
- Do not use **Environment.Exit()** to terminate the program.
- Do not change the given code template.

## Question 5

"SmartHealth, a healthcare management platform, provides an interface for patients to input their health information. This system ensures the accuracy and security of the data entered. Upon validation, the system calculates health insurance premiums based on the user inputs. Robust exception handling is implemented to manage any erroneous inputs effectively.

Consider the following scenario:

You receive input including the patient's name, age, gender, height, weight, blood pressure, and cholesterol levels in the UserInterface class.

Component Specification: **HealthDataProcessor** (Utility class)

| Type (Class) | Method | Parameters | Responsibilities |
|---|---|---|---|
| HealthDataProcessor | validatePatientDetails | int age, String gender, double height, double weight, int bloodPressure, double cholesterol | This method validates the patient details according to specified rules. If all details are valid, it returns true. If any detail is invalid, it throws an InvalidHealthDataException with the appropriate error message. |

***<u>Validation Rules:</u>***

- **Age**: Must be between 18 and 100 inclusive. If not, throw an InvalidHealthDataException with the message "Invalid age".
- **Gender**: Must be either 'Male' or 'Female' (case sensitive). If not, throw an InvalidHealthDataException with the message "Invalid gender".
- **Height**: Must be greater than 0. If not, throw an InvalidHealthDataException with the message "Invalid height".
- **Weight**: Must be greater than 0. If not, throw an InvalidHealthDataException with the message "Invalid weight".
- **Blood Pressure:** Must be within the range of 60 to 220 mmHg. If not, throw an InvalidHealthDataException with the message "Invalid blood pressure".
- **Cholesterol:** Must be greater than or equal to 0. If not, throw an InvalidHealthDataException with the message "Invalid cholesterol".

HealthDataProcessor calculateInsurancePremium double height, double weight, int age, int bloodPressure, double cholesterol This method calculates and returns the health insurance premium based on the patient's health metrics. Guidelines for premium calculation are based on BMI, blood pressure, and cholesterol levels, with specific rates for different age groups and genders.

Component Specification: **InvalidHealthDataException** (This class inherits the Exception Class)

| Type (Class) | Responsibilities |
|---|---|
| InvalidHealthDataException | Provided with a single-argument constructor: public InvalidHealthDataException(String message). It is thrown when age, gender, height, weight, blood pressure, or cholesterol levels do not follow the validation rules. |

The main method in the UserInterface class starts by getting the patient's name, age, gender, height, weight, blood pressure, and cholesterol levels, and then calls the validatePatientDetails() method to validate these details. If the patient details are valid, the method calculates the insurance premium using the calculateInsurancePremium() method and prints the result. If any validation fails, an InvalidHealthDataException is caught, and the exception message is displayed.

Note:

- Propagate the exceptions that occur in the HealthDataProcessor class and handle them in the main method.
- In the sample input and output provided, the highlighted text in bold corresponds to the input given by the user, and the rest of the text represents the output.

calculateInsurancePremium() method (take some sample value and calculate the premium )
**Example**

if height,weight,bloodpressue and cholesterol very high , premium 1500

if height,weight,bloodpressue and cholesterol high , premium 1000

if height,weight,bloodpressue and cholesterol normal , premium 750

## Question 6

A Travel Agency, operate a flight booking portal, allowing customers to book flight tickets. The Agency needs a system to manage flight booking efficiently. This system should allow customers to book tickets, view their booking details, and recalculate the total price of bookings. Help them wring your C* skills

Functional Requirements:

In Program class,

public static ArrayList Bookings { get; set; } = new ArrayList () ;

"This in already provided in the code template.

Implement the following. public Properties in Booking class

| Type | Property Name |
|---|---|
| string | Name |
| string | Price |
| string | Destination |
| string | Seat Number |

Requirements Description

1. Add booking details to the property.

2. Find booking, details by booking, ID!

3. Calculate the total piece of all bookings.

| class Name | Method Name | Parameters | Description |
|---|---|---|---|
| Booking Uility | AddBookings | int id. Booking, bookings | This method adds the booking details<br><br>It first adds the unique booking. ID, then adds the Corresponding bookings. details to the next index, in the Bookings Arraylist properties in the Program Class |

| | | | |
|---|---|---|---|
| | | | This method returns void |
| Booking Utility | FindBooking | int id | This method returns the booking. detail as a Booking object!<br><br>. When ID is found it should return the booking, details in the next index of Id<br>. When ID in not found, it should return null<br><br>Refer to sample input and output |
| Booking Utility | Get Total Price | - | This method Calculates and returns the total price of all bookings as a double value |

In the Program class, implement the below given method

Method 1: public static void Main (string[] args)

Description 1

1. Prompt the user to enter the choice from the list of options shown.
2. For choice 1, prompt the user to enter the number of bookings and get the detail of the booking from the user call the Add Bookings method, and pass the unique Booking Id ( Booking Id needs to start from 101 and so on.) and booking, details to add them to the Bookings Arraylist property in the Program Class.
3. For choice 2, prompt the user to enter the unique booking Id for finding the booking detail. call the Find Booking method and pass the Id. If the Id in present, print the result in this format {Name} - {Price} - {Destination"} { seat Number }". It this method returns null, print "Booking Id not found'
4. For choice 3, call the GetTotalPrice method, to find the total price and print "Total price for booking In {TotalPrice} rupees
5. For choice 4, Exit the program and print "Enjoy your journey".

Note: Output in Cave - Sensitive

Refer to sample input and output.

Note

Keep

all methods public

Do not use Environment. Exit () to terminate the program.

Do not change the given code template.

Sample Input/Output 1

```
Welcome to flight booking, portal
1. Book the ticket
2. Find the booking details
3. Find the total price
4. Exit
Enter your choice
1
Enter the number of booking
2
Enter the price
25000
Enter the destinations
New York
Enter the seat number
A321

Ticket Booked
```

Sample Input/Output 2

```
Welcome to flight booking, portal
1. Book the ticket
2. Find the booking details
3. Find the total price
4. Exit
```

```
Enter the choice
2
Enter the id to find the booking details
102

Paul – 35000 – Sydney – B453
```

Sample Input/Output 3

```
Welcome to flight booking, portal
1. Book the ticket
2. Find the booking details
3. Find the total price
4. Exit

Enter the choice
2
Enter the id to find the booking details
104

Booking id not found
```

Sample Input/Output 4

```
Welcome to flight booking, portal
1. Book the ticket
2. Find the booking details
3. Find the total price
4. Exit

Enter the choice
3
```

```
Total price for booking is 60000 rupees
```

.

Sample Input/Output 4

```
Welcome to flight booking, portal

1. Book the ticket

2. Find the booking details

3. Find the total price

4. Exit


Enter the choice

4


Enjoy your journey
```

## Question 7

As per Sam's instructions, John has also started working as a freelancer and got a project from a mobile shop. The project requires John to manage mobile details, such as model, brand, and price, and to group the mobiles by their brand. The application will help the company manage their mobile inventory more efficiently, allowing them to keep track of their stock and prices accurately.

Help him create an application that meets the requirements by using your C# skills.

**Functionalities:**

In class **Program,**

**public static SortedDictionary<int, Mobile> mobileDetails - **this sorted dictionary is already provided.

In class **Mobile**, implement the below properties.

| **Data Type** | **Property Name** |
|---|---|
| string | Model |
| string | Brand |
| int | Price |

In class **MobileUtility**, implement the below methods.

| **Method** | **Description** |
|---|---|
| public void AddMobileDetails(string model, string brand, int price) | This method is used to add the mobile details, such as model, brand and price passed as an argument to the **mobileDetails** dictionary.<br><br>The key for the dictionary is set to be one more than the current number of items in the dictionary.<br><br>Now the current number of items in the dictionary is **0**. |

| | |
|---|---|
| public SortedDictionary<string, List<Mobile>> GroupMobilesByBrand() | This method is used to group the mobiles by their brand, add them to the dictionary, and return them. |

In **Program** class - **Main** method,

1. Get the values from the **user**.
2. Call the methods accordingly and display the result as per the Sample Output.

**Note:**

- Keep the methods and classes as **public**.
- Please read the method rules **clearly**.
- Do not use **Environment.Exit()** to terminate the program.
- Do not change the given code **template**.

**Summary :**

Through working on this application, he has learned about the generic collection **Sorted Dictionary** class in C#.

**Sorted Dictionary** is a generic collection class in C# that represents a collection of key/value pairs sorted by key.

**Sample Input/Output:**

1. Add Mobile Details

2. Group Mobiles By Brand

3. Exit

Enter your choice

**1**

Enter the model

**Nokia 7.2**

```
Enter the brand
Nokia
Enter the price
10000

Mobile details added successfully

1. Add Mobile Details
2. Group Mobiles By Brand
3. Exit
Enter your choice
1
Enter the model
J7 nxt
Enter the brand
Samsung
Enter the price
11000

Mobile details added successfully


1. Add Mobile Details
2. Group Mobiles By Brand
3. Exit
Enter your choice
1
Enter the model
Sony Xperia Pro
Enter the brand
```

```
Sony
Enter the price
25000

Mobile details added successfully

1. Add Mobile Details
2. Group Mobiles By Brand
3. Exit
Enter your choice
1
Enter the model
Sony Xperia 1
Enter the brand
Sony
Enter the price
20000

Mobile details added successfully

1. Add Mobile Details
2. Group Mobiles By Brand
3. Exit
Enter your choice
1
Enter the model
Samsung Note 4
Enter the brand
Samsung
```

```
Enter the price
14000
Mobile details added successfully

1. Add Mobile Details
2. Group Mobiles By Brand
3. Exit
Enter your choice
2
```

| |
|---|
| Nokia |
| Nokia 7.2 |

| |
|---|
| Samsung |
| J7 nxt<br>Samsung Note 4 |

| |
|---|
| Sony |
| Sony Xperia Pro<br>Sony Xperia 1 |

```
1. Add Mobile Details
2. Group Mobiles By Brand
3. Exit
Enter your choice
3
Thank you
```

## Question 8

Sam's uncle owns a hospital known for its exceptional treatment options for various health conditions. As a priority, they ensure timely and efficient care for all patients. To maintain accurate and up-to-date records, a system is needed to add new patient information upon admission and remove it upon discharge.

Utilising your skills in C# programming, assist them by creating and implementing a system to efficiently manage patient records at Sam's uncle's hospital.

**Functionalities:**

In class **Program**, implement the below given **public static** property.

| **Data Type** | **Property Name** |
|---|---|
| Queue<Patient> | PatientQueue |

In class **Patient,** implement the below given properties and methods.

| **Data Type** | **Property Name** |
|---|---|
| int | Id |
| string | Name |
| string | Address |

| **Method** | **Description** |
|---|---|
| public Queue<Patient> AddPatientDetails(int id, string name, string address) | This method is used to add the patient details, such as id, name and address to the **PatientQueue** and return it. |
| public string GetPatientDetails() | This method is used to get the patient details from the **PatientQueue.**<br><br>In this method, get the first admitted patient details from **PatientQueue** and return the details as a string by seperating it by one space in-between. |

| | |
|---|---|
| | **Format :**<br>**Id" "Name" "Address**<br>**" "** determines One space |
| public Queue<Patient> RemovePatientDetails() | This method is used to remove the patient details from the **PatientQueue**.<br>In this method, remove the first admitted patient details from **PatientQueue** and return the remaining patient details. |

In **Program** class - **Main** method,

1. Get the values from the **user**.
2. Call the methods accordingly and display the result as per the Sample Output.

**Note:**

- Keep the methods and classes as **public**.
- Please read the method rules **clearly**.
- Do not use **Environment.Exit()** to terminate the program.
- Do not change the given code **template**.

## Question 9

John is an avid reader who constantly acquires new books. He organises his collection on his home shelves, arranging them in chronological order of acquisition, but he occasionally disarrays them. Utilise your C# programming skills and help John maintain a proper and organised book collection by developing a programme that tracks the acquisition date of each book and sorts them accordingly on the shelves. The programme could also include a feature that allows John to easily rearrange the books if he wishes to change the order.

**Functionalities:**

In class **Program**, implement the below given **public static** property.

| Data Type | Property Name |
|---|---|
| Stack<Book> | BookStack |

In class **Book,** implement the below given properties and methods.

| Data Type | Property Name |
|---|---|
| int | Id |
| string | Name |
| string | Author |

| Method | Description |
|---|---|
| public Stack<Book> AddBookDetails(int id, string name, string author) | This method is used to add the book details, such as id, name and author to the **BookStack** and return it. |
| public string GetBookDetails() | This method is used to get the book details from the **BookStack.** |

| | |
|---|---|
| | In this method, get the recently added book details from **BookStack** and return them as a string by separating it by one space in-between.<br><br>**Format :**<br>**Id" "Name" "Author**<br>" " determines One space |
| public Stack<Book> RemoveBookDetails() | This method is used to remove the book details from the **BookStack** .<br>In this method, remove the recently added book details from **BookStack** and return the remaining book details. |

In **Program** class - **Main** method,

1. Get the values from the **user**.
2. Call the methods accordingly and display the result as per the Sample Output.

**Note:**

- Keep the methods and classes as **public**.
- Please read the method rules **clearly**.
- Do not use **Environment.Exit()** to terminate the program.
- Do not change the given code **template**.

**Summary :**

Through working on this task, we have learned about generic collection **Stack** class in C#.

**Stack** is a generic collection class that represents a last-in, first-out (LIFO) collection of objects.

## Question 10

Sam and John's studying institute is updating its course offerings and fees. They need an application to add new courses with their respective fees and also remove existing courses. The application also has a feature to display a list of current courses with their respective fees.

Help them to create an application by using your C# skills.

**Functionalities:**

In class **Program**, implement the below given **public static** property.

| **Data Type** | **Property Name** |
|---|---|
| Dictionary<string, int> | CourseDetails |

In class **Course,** implement the below methods.

| **Method** | **Description** |
|---|---|
| public void AddCourseDetails(string name, int fee) | This method is used to add the course details, such as course name and fee, to the dictionary **CourseDetails**. |
| public void RemoveCourseDetails(string name) | This method is used to remove the course details based on the name passed as an argument from the dictionary **CourseDetails**. |
| public Dictionary<string, int> SortCourseByFee() | This method is used to sort the course details based on the fee in ascending order and return that as a **dictionary**. |

In **Program** class - **Main** method,

1. Get the values from the **user**.
2. Call the methods accordingly and display the result as per the Sample Output.

**Note:**

- Keep the methods and classes as **public**.
- Please read the method rules **clearly**.

- Do not use **Environment.Exit()** to terminate the program.
- Do not change the given code **template**.

**Summary :**

Through working on this application, they have learned about the generic collection **Dictionary class** in C#.

**Dictionary class** in C# is a generic collection that stores key-value pairs where the key is unique and used to access the corresponding value.

## Question 11

In that institute, the trainer is instructed to maintain perfect attendance for every single student. Since few students are moved out, now he has to remove their names and sort the attendance in alphabetical order using an array list.

Help him to complete the task by using your C# skills.

**Functionalities:**

In class **Program**,

**public static ArrayList Attendance = new ArrayList() -** this ArrayList is already provided.

implement the below-given methods.

| **Method** | **Description** |
|---|---|
| public bool RemoveStudent(String name) | This method is used to remove the student from array list based on the name passed as an argument.<br>If this method removes the student, then return **true**. Otherwise, return **false**. |
| public void SortTheAttendance() | This method is used to sort the names present in the array list in ascending order. |

ArrayList **Attendance** already contains the following names:

John

Peter

Jacob

Archie

Sophie

Veronica

Elizabeth

Charles

In **Program** class - **Main** method,

**1.** Get the values from the **user**.

**2.** Call the methods accordingly and display the result as per the Sample Output. If the RemoveStudent method returns true, then display **Removed successfully**. else, display **Remove failed**

## Question 12

**Scenario:**

The proprietor of a vintage automobile dealership is offering a Halloween promotion for their inventory, with discounts varying based on the body style of the vehicle. For SUVs, a **10%** discount will be applied, while Sedans will receive a **25%** discount. They approached John, a developer who wants to create an application to calculate the final price of a vehicle after the promotional discount has been applied.

Help him to create an application by using your C# coding skills.

**Functionalities:**

In class **Owner,** declare the below given variable.

| **Field** | **Access specifier** |
|---|---|
| String ownerName | protected |

In class **Car,** implement the below given fields, properties and methods. The class **Car** should inherit the class **Owner.**

| **Field** | **Access specifier** |
|---|---|
| double price | internal |
| String bodyStyle | private |

| **Property** | **Access specifier** |
|---|---|
| String BodyStyle | public |

| Method | Description |
|---|---|
| public bool ValidateBodyStyle(string bodyStyle) | In this method, pass the body style as an argument.<br>If the body style is **SUV** or **Sedan** then return **true**.<br>If the body style is other than this return **false**. |
| public double CalculatePrice() | In this method,<br>-- If the body style is **SUV** then discount 10% to the price,<br>-- If the body style is **Sedan** then discount 25% to the price<br>Store and return the result as a **double** data type.<br>-- If the body style is other than this, return **0.** |
| public void SetOwnerName(string name) | In this method, pass the name as an argument.<br>Set the name to the field **ownerName.** |

**Note : SUV** or **Sedan (Case-Sensitive)**

In **Program,** class **Main** method,

1. Get the **input** values from the user.

2. Call the **ValidateBodyStyle** method. If it returns false, then display the message **"Invalid Car Type".** If it returns true, then move on to step 3.

3. Call the **CalculatePrice** method and then call the **SetOwnerName** method.

4. Use the values in the methods and display the output as shown in the sample output.

**Note:**

- Keep the methods and class as **public**.
- Please read the method rules **clearly**.
- Do not use **Environment.Exit()** to terminate the program.
- Do not change the given code template.

## Question 13

**Functionalities:**

In class **Registration,** implement the below-given **methods.**

| **Method** | **Description** |
|---|---|
| public bool ValidateMobileNumber(string mobileNumber) | This method is used to validate the mobile number.<br>If the mobile number is valid, then return **true**. Otherwise, return **false**.<br>Valid Pattern : **xxxxxxxxxx** or **+xx xxxxxxxxxx**<br>**x** denotes numbers.<br>**Note :** You must use **Regular expression.** |
| public double CalculateRegistrationFee(string category, double baseFee) | This method is used to calculate the registration fee of the event based on the category and return it.<br>If the category is **Student**, then give **20%** discount to the base fee.<br>If the category is **Working Professional** , then give **20%** extra amount to the base fee. |

**Note :** category is case sensitive.

In **Program** class - **Main** method,

**1.** Get the values from the **user** as per the Sample Input.

**2.** Call the ValidateMobileNumber method, if this method returns true, then move on to step 3. Otherwise, display **Please enter valid mobile number.**

**3.** Call the method CalculateRegistrationFee and display the result as per the Sample Output.

## Question 14

A new grocery store has recently opened in a residential area. The owner has invested significant effort into setting up the shop and stocking it with various grocery items. The store is now ready to serve customers. However, managing a grocery store can be a challenging and time-consuming task, as there are numerous details to keep track of, such as inventory and billing. In order to streamline these processes and make the store run more efficiently, the owner decides that they need a software solution.

As their software consultant, you have the expertise to develop a C# application that can help them manage these details and improve the overall operation of the grocery store.

**Functionalities:**

In the class ItemDetails, implement the below-given properties.

**Class: ItemDetails**

**Properties:**

- string ItemName
- int Quantity
- double PricePerUnit
- double TotalPrice
- double Discount

**Class: Billing (Inherits ItemDetails)**

**Methods:**

1. **public bool ValidateItemName(string itemName)**:
    - This method is used to validate the item name.
    - It takes itemName as a parameter.
    - If the itemName is "Apple", "Banana", or "Orange", return true.
    - Otherwise, return false and print "Invalid item name" in the Main method.
2. **public ItemDetails GenerateBill()**:
    - This method is used to calculate the total cost and discount details for the item.
    - Using Quantity and PricePerUnit values present in the ItemDetails class, calculate the TotalPrice and Discount and then store the result in an ItemDetails object and return it.
    - To calculate TotalPrice: TotalPrice = Quantity * PricePerUnit

## Condition:

```
if (TotalPrice >= 50 && TotalPrice <= 200)
{
   Discount = TotalPrice * 0.05;
}
else if (TotalPrice > 200 && TotalPrice <= 500)
{
   Discount = TotalPrice * 0.10;
}
else if (TotalPrice > 500)
{
   Discount = TotalPrice * 0.15;
}
else
{
   Discount = 0;
}
```

## Question 15

A Travel Agency operates a flight booking portal, allowing customers to book flight tickets. The agency needs a system to manage flight bookings efficiently. This system should allow customers to book tickets, view their booking details, and calculate the total price of all bookings. Help them using your C# skills.

**Functional Requirement:**

In **Program** class,

public static ArrayList Bookings { get; set; } = new ArrayList() - This is **already provided** in the code template.

Implement the following public Properties in **Booking** class

| **Type** | **Property Name** |
|---|---|
| string | Name |
| double | Price |
| string | Destination |
| string | SeatNumber |

| **Req. #** | **Requirements Description** | **Class Name** | **Method Name** | **Parameters** | **Description** |
|---|---|---|---|---|---|
| **1** | Add booking details to the property | **BookingUtility** | AddBookings | int id, Booking bookings | This method **adds** the booking details.<br><br>It first adds the **unique booking ID**, then adds the corresponding booking details to the next index in the **Bookings** ArrayList property in |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | the Program Class.<br><br>This method returns **void**. |
| **2** | Find booking details by booking ID | **BookingUtility** | FindBooking | int id | This method returns the **booking details** as a Booking object.<br><br>When the ID is **found**, it should returnthe booking details in the next index of Id.<br><br>When the ID is **not found**, it should return **null**.<br><br>**Refer to sample input and output.** |
| **3** | Calculate the total price of all bookings | **BookingUtility** | GetTotalPrice | - | This method calculates and returns the **total price** of all bookings as a double value. |

| **S. No.** | In the **Program** class, implement the below-given **Methods**. | |
|---|---|---|
| **1.** | **Method1:** | public static void Main(string[] args) |
| | **Description1:** | 1) Prompt the user to enter the choice from the list of options shown.<br><br>2) For choice **1**, prompt the user to enter the **number of bookings** and get the details of the booking from the user, call the |

| | | |
|---|---|---|
| | | **AddBookings** method, and pass the unique Booking Id (Booking **Id** needs to start from **101** and so on.) and booking details to add them to the **Bookings** ArrayList property in the Program Class.<br><br>3) For choice **2**, prompt the user to enter the unique Booking Id for finding the booking details. Call the **FindBooking** method and pass the Id. If the Id is **present**, print the result in this format **"{Name} - {Price} - {Destination} - {SeatNumber}".** If this method returns **null**, print **"Booking Id not found"**<br><br>4) For choice **3**, call the **GetTotalPrice** method, to find the total price and print **"Total price for booking is {TotalPrice} rupees".**<br><br>5) For choice **4**, Exit the program and print **"Enjoy your journey".**<br><br>**Note:** Output is **Case-Sensitive.**<br><br>**Refer to Sample input and output.** |

**Note:**

- Keep **all methods** public.

- Do not use **Environment.Exit()** to terminate the program.

- Do not change the given code template.

- In the Sample Input / Output provided, the highlighted text in bold corresponds to the input given by the user, and the rest of the text represents the output.

- Input should only be gathered in the **Main method.**

**Sample Input/Output 1:**

Welcome to flight booking portal

1. Book the tickets

2. Find the booking details

3. Find the total price

4. Exit

Enter your choice

**1**

Enter the number of bookings

**2**

Enter the passenger 1 name

```
John
Enter the price
25000
Enter the destination
New York
Enter the seat number
A321

Ticket Booked

Enter the passenger 2 name
Paul
Enter the price
35000
Enter the destination
Sydney
Enter the seat number
B453

Ticket Booked

1. Book the tickets
2. Find the booking details
3. Find the total price
4. Exit
Enter your choice
2
Enter the id to find the booking details
102
```

```
Paul - 35000 - Sydney - B453

1. Book the tickets
2. Find the booking details
3. Find the total price
4. Exit
Enter your choice
2
Enter the id to find the booking details
104

Booking Id not found

1. Book the tickets
2. Find the booking details
3. Find the total price
4. Exit
Enter your choice
3

Total price for booking is 60000 rupees

1. Book the tickets
2. Find the booking details
3. Find the total price
4. Exit
Enter your choice
4

Enjoy your journey
```