1.Power Managements

Froma is a well-known power bank import and export company that specialises in providing high-quality power banks to customers worldwide. The company's management team has recently realised that there is a growing demand for power banks with higher battery capacities.

Additionally, the team wants software to develop a marketing strategy to promote the power banks to customers and help them understand the benefits of having a power bank with a high battery capacity.

You being their software consultant help them by developing a C# application.

Functionalities:

In class **Program,** implement the below-given method.

public static Dictionary<string, int> PowerBankDetails -In the code template, it is already provided.

implement the features listed below.

Method	Description
public void AddPowerBankDetails (string[ ] powerBank)	This method is used to add the power bank details into the **PowerBankDetails** dictionary. This method should separate the values in each string by a **colon(:)** from the input array and store them in a **Dictionary**.
public int FindBatteryPower(String powerBankName)	This method is used to find the battery power of the power bank based on the **powerBankName** passed as an argument. If the power bank name is found in the Dictionary, then return that **battery power**. else, return **-1** and print "**No power banks are available**" in the Main method.
public List<String> FindTheHighestPowerBattery()	This method is used to find the highest capacity battery power bank from the **PowerBankDetails** dictionary, then store the power bank name as a **list** and return it. **Note:** If the highest capacity battery has more than one power bank, then consider that all are the highest capacity battery power banks and they need to be added to the list.

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

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

2. Call the methods accordingly and display the result.

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

Note:

- Keep the method 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.

Sample Input and Output :

1. Add Power Bank Details

2. View Battery Power

3. View Power Banks With Highest Battery Power

4. Exit

Enter the choice

1

Enter the number of entries

5

Hero Cell:30000

Bull Cell:40000

Ivp Cell:70000

Tvp Cell:20000

Netron Cell:70000

1. Add Power Bank Details

2. View Battery Power

3. View Power Banks With Highest Battery Power

4. Exit

Enter the choice

2

Enter the power bank name needs to be searched

K34

No power banks are available

1. Add Power Bank Details

2. View Battery Power

3. View Power Banks With Highest Battery Power

4. Exit

Enter the choice

2

Enter the power bank name needs to be searched

Hero Cell

30000

1. Add Power Bank Details

2. View Battery Power

3. View Power Banks With Highest Battery Power

4. Exit

Enter the choice

3

Power Banks with the highest battery power are:

Ivp Cell

Netron Cell

1. Add Power Bank Details

2. View Battery Power

3. View Power Banks With Highest Battery Power

4. Exit

Enter the choice

4

Thank you.

2. **E-insurance**

E-insurance is a well-known e-service shop in the city that specialises in offering government insurance schemes to its customers. They have a wide variety of plans available, each with its own unique benefits and coverage options.

Recently, the management at E-insurance has decided to improve their customer service by implementing a new system that allows customers to easily find out the monthly amount for a specific scheme based on its name.

You being their software consultant help them by developing a C# application.

Functionalities:

In class **Program,** implement the below-given method.

public static Dictionary<string, double> SchemeDetails -In the code template, it is already provided.

implement the features listed below.

Method	Description
public void AddSchemeDetails (string[ ] scheme)	This method is used to add the scheme details into the **SchemeDetails** dictionary. This method should separate the values in each string by a **colon(:)** from the input array and store them in a **Dictionary**.
public double FindSchemeMonthlyAmount (String schemeName)	This method is used to find the monthly amount of the scheme based on the **schemeName** passed as an argument. If the scheme name is found in the Dictionary, then return that scheme **amount**. else, return **-1** and print "**No schemes are available**" in the Main method.
public List<string> FindLowestMonthlyAmountScheme()	This method is used to find the lowest amount scheme from the **SchemeDetails** dictionary, then store the scheme name as a **list** and return it. **Note:** If the lowest amount has more than one scheme, then consider that all are the lowest amount scheme and they need to be added to the list.

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

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

2. Call the methods accordingly and display the result.

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

Note:

- Keep the method 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.

Sample Input and Output :

1. Add Scheme Details

2. View Monthly Amount Based on Name

3. View Schemes With Lowest Monthly Amount

4. Exit

Enter the choice

1

Enter the number of entries

5

The Ayushman Bharat:5200

National Pension System:6000

The Employees' State Insurance Scheme:8000

The Central Government Health Scheme:4800

The Pradhan Mantri Suraksha Bima Yojana:4800

1. Add Scheme Details

2. View Monthly Amount Based on Name

3. View Schemes With Lowest Monthly Amount

4. Exit

Enter the choice

2

Enter the scheme name needs to be searched

Atal Pension Yojana

No schemes are available

1. Add Scheme Details

2. View Monthly Amount Based on Name

3. View Schemes With Lowest Monthly Amount

4. Exit

Enter the choice

2

Enter the scheme name needs to be searched

The Employees' State Insurance Scheme

Amount is : 8000

1. Add Scheme Details

2. View Monthly Amount Based on Name

3. View Schemes With Lowest Monthly Amount

4. Exit

Enter the choice

3

Schemes with the lowest monthly amount are:

The Central Government Health Scheme

The Pradhan Mantri Suraksha Bima Yojana

1. Add Scheme Details

2. View Monthly Amount Based on Name

3. View Schemes With Lowest Monthly Amount

4. Exit

Enter the choice

4

Thank you.

3.E-Restaur

A new E-Restaur has recently opened in a bustling commercial area. The owner has put a lot of effort into setting up the shop and hiring a team of employees and is eager to start serving customers.

However, they quickly realize that managing a E-Restaur can be a challenging and time-consuming task, as there are numerous details to keep track of, such as inventory, orders, and customer information. In order to streamline these processes and make the shop 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 restaurant.

Functionalities:

In class **FoodDetails,** implement the below-given properties.

Class	Properties
FoodDetails	string FoodType int Quantity int PricePerPiece double TotalPrice double Discount

In class **Billing,** implement the below-given methods and also **Inherit** the class **FoodDetails**.

Method	Description
public bool ValidateFoodType(string foodType)	This method is used to validate the food type. In this method we pass **foodType** as a parameter. If the foodType is **Samosa** or **Spring Roll** or **Empanada**, then return **true**. Otherwise return **false**, print **Invalid food type** in Main method
public FoodDetails GenerateBill()	This method is used to calculate the cost of food and discount details for that food. using **Quantity** and, **PricePerPiece** values present in the class **FoodDetails,** calculate the price and discount and then store the result in **FoodDetails** object and return it. To calculate **TotalPrice**, **TotalPrice** = **Quantity** * **PricePerPiece** To calculate **discount price,** please refer the below table. **Discount = TotalPrice * Discount %** Then store the details such as **FoodType**, **Quantity** , **PricePerPiece** , **TotalPrice**, and **Discount** in object **FoodDetails** and return it.

Note:

FoodType is **Case-sensitive**.

Calculation for Discount Price :

TotalPrice	Discount Percentage
>=100 and < =500	10%
> 500 and <= 1000	15%
>1000	20%
<100	0%

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

1. Get the **FoodType**, **Quantity**, and **PricePerPiece** value from the **user**.

2. Call the **ValidateFoodType** method and if it returns true, then move on to step 3, otherwise print **Invalid food type** in the Main method

3. Use the values in method **GenerateBill** 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.

Sample Input 1:

Enter the food type

Samosa

Enter the quantity

20

Enter the price per piece

40

Sample Output 1:

FoodType	Quantity	PricePerPiece	Discount	TotalPrice
Samosa	20	40	120	800

Sample Input 2:

Enter the food type

Empanada

Enter the quantity

1

Enter the price per piece

70

Sample Output 2:

FoodType	Quantity	PricePerPiece	Discount	TotalPrice
Empanada	1	70	0	70

Sample Input 3:

Enter the food type

Pizza

Enter the quantity

5

Enter the price per piece

150

Sample Output 3:

Invalid food type

5.Hill Adventure

Hamels is an experienced traveller and an avoid hilleer. He has decided that his next adventure will be climbing a hill and he wants to choose one that is particularly tall. He has researched several hills and narrowed down his options to a few that he is interested in climbing.

However, he wants to make sure that he chooses the tallest one out of the options he has. He wants software that provides information on the height of different hills around the world.

You being his software consultant help him by developing a C# application.

Functionalities:

In class **Program,** implement the below-given method.

public static Dictionary<string, int> HillDetails -In the code template, it is already provided.

implement the features listed below.

Method	Description
public void AddHillDetails (string[ ] hill)	This method is used to add the hill details into the **HillDetails** dictionary. This method should separate the values in each string by a **colon(:)** from the input array and store them in a **Dictionary**.
public int FindHillHeight (String hillName)	This method is used to find the height of the hill based on the **hillName** passed as an argument.

	If the hill name is found in the Dictionary, then return that hill **height**. else, return **-1** and print "**No hills are available**" in the Main method.
public List<String> FindTheHighestHills()	This method is used to find the highest height hill from the **HillDetails** dictionary, then store the hill name as a **list** and return it. **Note:** If the highest height has more than one hill, then consider that all are the highest height hills, and they need to be added to the list.

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

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

2. Call the methods accordingly and display the result.

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

Note:

- Keep the method 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.

Sample Input and Output :

1. Add Hill Details

2. View Hill Height

3. View Hills With Highest Height

4. Exit

Enter the choice

1

Enter the number of entries

5

Mount Elbrus:7970

Aconcagua:7827

Manaslu:4510

Mount Vinson:7970

K2:6120

1. Add Hill Details

2. View Hill Height

3. View Hills With Highest Height

4. Exit

Enter the choice

2

Enter the hill name needs to be searched

K34

No hills are available

1. Add Hill Details

2. View Hill Height

3. View Hills With Highest Height

4. Exit

Enter the choice

2

Enter the hill name needs to be searched

Aconcagua

Height is : 7827

1. Add Hill Details

2. View Hill Height

3. View Hills With Highest Height

4. Exit

Enter the choice

3

Hill names with the highest height are:

Mount Elbrus

Mount Vinson

1. Add Hill Details

2. View Hill Height

3. View Hills With Highest Height

4. Exit

Enter the choice

4

Thank you.

6.Scenario

6.Wings APP

Fly Wings, a famous bird shop in the city, wants to have a software application that allows them to quickly and easily check the number of remaining birds based on the bird's name. The manager has approached you, a software developer, to help with this task. The application will need to be able to track the number of birds in stock, their types, and prices. It should also be able to update the inventory when birds are sold or new ones are added.

Functionalities:

In class **Program,** implement the below-given method.

public static Dictionary<string, int> BirdDetails -In the code template, it is already provided.

implement the features listed below.

Method	Description
public void AddBirdDetails (string[ ] bird)	This method is used to add the birds details into the **BirdDetails** dictionary. This method should separate the values in each string by a **colon(:)** from the input array and store them in a **Dictionary**.
public int FindTheBirdCount (String birdName)	This method is used to find the count of the birds based on the **birdName** passed as an argument. If the bird name is found in the Dictionary, then return that bird **count**. else, return **-1** and print "**No birds are available**" in the Main method.
public List<String> FindTheHighestCountOfBird**()**	This method is used to find the highest count birds from the **BirdDetails** dictionary, then store the birds name as a **list** and return it. **Note:** If the highest count has more than one bird, then consider that all are the highest count birds, and they need to be added to the list.

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

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

2. Call the methods accordingly and display the result.

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

Note:

- Keep the method 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.

Sample Input and Output :

1. Add Bird Details

2. View Number of Birds By Bird Name

3. View Birds With Highest Count

4. Exit

Enter the choice

1

Enter the number of entries

5

Dove:85

Owl:23

Love bird:32

Penguin:50

Robin:85

1. Add Bird Details

2. View Number of Birds By Bird Name

3. View Birds With Highest Count

4. Exit

Enter the choice

2

Enter the bird name to get the number of birds

Bat

No birds are available

1. Add Bird Details

2. View Number of Birds By Bird Name

3. View Birds With Highest Count

4. Exit

Enter the choice

2

Enter the bird name to get the number of birds

Owl

Number of Birds : 23

1. Add Bird Details

2. View Number of Birds By Bird Name

3. View Birds With Highest Count

4. Exit

Enter the choice

3

Bird names with the highest count are:

Dove

Robin

1. Add Bird Details

2. View Number of Birds By Bird Name

3. View Birds With Highest Count

4. Exit

Enter the choice

4

Thank you.

7.Property Tax Management

The Property Tax Department is seeking to streamline its operations and has decided to outsource the tax calculation process to Info Tech Software Company. As a software consultant with expertise in C# development, you have been approached to create a custom application to handle this important task. The goal of this project is to design and develop a C#-based solution that accurately calculates the property tax for each property owner in an efficient and reliable manner.

Functionalities:

In class **PropertyTax,** implement the below-given properties.

Class	Properties

PropertyTax	string PlotNumber string OwnerName string BuildingType int SquareFeet

In class **Service,** implement the below-given methods and also **Inherit** the class **PropertyTax**.

Method	Description
public bool ValidatePlotNumber(string plotNumber)	This method is used to validate the plot number. The plot number should consists of 10 characters. The first three characters should be **alphabets (lower case)**, followed by one slash **(/)**, followed by 6 characters which should be **digits**.. If plot number is valid then return true. Otherwise return **false.** **Note:** PlotNumber is **Case-sensitive**.
public double CalculateTaxAmount()	This method is used to calculate the tax amount based on building type and return the amount. To calculate the tax amount refer the below-given table

Building Type	Tax Amount/sqFeet(in Rs)
Commercial	1.5
Residential	0.75

Note: Building Type is **Case-sensitive**.

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

1. Get the **PlotNumber**, **OwnerName, BuildingType and SquareFeet** value from the **user**.

2. Call the **ValidatePlotNumber** method and if it returns true, then move on to next step. else display **Invalid plot number**

3. Use the values in method **CalculateTaxAmount** and display the value 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.

Sample Input/Output1:

Enter the plot number

set/909090

Enter the owner name

James

Enter the building type

Commercial

Enter the square feet

2000

Tax Amount: 3000

Sample Input/Output2:

Enter the plot number

454/909090

Enter the owner name

Alex

Enter the building type

Residential

Enter the square feet

2000

Invalid plot number

8. **Rating Grade**

Alex is a medical student who wants to calculate his GPA for the current semester. He obtained different marks in each subject. He wants to find his overall GPA as well as the grade that corresponds to that GPA. The credit point for each subject is 3. He needs software to calculate the GPA and the grade as well.

As his software consultant, you can help him by developing a C# application.

Functionalities:

In class **Program,** implement the below-given method.

public static List<int> MarkList -In the code template, it is already provided.

implement the features listed below.

Method	Description
public void AddMarks(int marks)	This method is used to add the **marks** into the **MarkList. marks** is passed as an argument.
public double GetGPAScored()	This method is used to find the **GPA** of all marks scored in the semester and return the value.

	GPA can be calculated based on the sum of products of each marks available in the **MarkList** and credit point for each subject, divide it by sum of credits. **Note:** The credit point of each subject is commonly **3**. GPA can be calculated based on the following formula : **GPA= (mark1 * 3) + (mark2 * 3) + ... + (markn * 3)/(List count * 3)** If the **List** is empty then return **-1** and print "**No Marks Available**" in **Main()** method.
public char GetGradeScored(double gpa)	This method is used to find the grade for **gpa** passed as an argument and return the grade. The grade point equivalent for each grade is mentioned in the below table. If the **gpa** less than **5** or greater than **10,** then return a null character and print "**Invalid GPA**" in **Main()** method.

GPA	Grade
Equal to 10	S
>= 9 and <10	A
>= 8 and <9	B
>= 7 and <8	C
>= 6 and <7	D
>= 5 and <6	E

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

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

2. Call the methods accordingly and display the result.

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

Note:

- Keep the method 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.

Sample Input and Output 1:

1. Add Mark

2. Calculate GPA

3. Get Grade

4. Exit

Enter your choice

1

Enter the mark scored

9

1. Add Mark

2. Calculate GPA

3. Get Grade

4. Exit

Enter your choice

1

Enter the mark scored

10

1. Add Mark

2. Calculate GPA

3. Get Grade

4. Exit

Enter your choice

1

Enter the mark scored

8

1. Add Mark

2. Calculate GPA

3. Get Grade

4. Exit

Enter your choice

1

Enter the mark scored

5

1. Add Mark

2. Calculate GPA

3. Get Grade

4. Exit

Enter your choice

2

GPA Scored: 8

1. Add Mark

2. Calculate GPA

3. Get Grade

4. Exit

Enter your choice

3

Grade Scored: B

1. Add Mark

2. Calculate GPA

3. Get Grade

4. Exit

Enter your choice

4

Thank You

Sample Input and Output 2:

1. Add Mark

2. Calculate GPA

3. Get Grade

4. Exit

Enter your choice

2

No Marks available

1. Add Mark

2. Calculate GPA

3. Get Grade

4. Exit

Enter your choice

4

Thank You

Sample Input and Output 3:

1. Add Mark

2. Calculate GPA

3. Get Grade

4. Exit

Enter your choice

1

Enter the mark scored

4

1. Add Mark

2. Calculate GPA

3. Get Grade

4. Exit

Enter your choice

2

GPA Scored: 4

1. Add Mark

2. Calculate GPA

3. Get Grade

4. Exit

Enter your choice

3

Invalid GPA

1. Add Mark

2. Calculate GPA

3. Get Grade

4. Exit

Enter your choice

4

9. **Grey Batteries**

Bright Batteries is a well-known battery shop in the centre of the city that specializes in the sale and replace of battery and other related devices. This type of shop typically carries a wide range of batteries from various brands, as well as replace for battery issues. They had difficulty counting sold batteries, and they require software to keep track of the details of their batteries.

As their software consultant, you can help them by developing a C# application.

Functionalities:

In class **Program,** implement the below-given method.

public static SortedDictionary<String, long> upsBatteryDetails -In the code template, it is already provided.

implement the features listed below.

Method	Description
public SortedDictionary<String,long> FindBatteryDetails(long soldCount)	This method is used to find the battery details by sold count. If the **soldCount** is available in the **upsBatteryDetails,** it should return that item as **SortedDictionary.** If the **soldCount** is not available in **upsBatteryDetails,** then return an empty **SortedDictionary.** If this method return empty **SortedDictionary,** then print "**Invalid sold count**" in **Main()** method.
public List<String> FindMinandMaxSoldBatteries()	This method is used to find the minimum and maximum sold batteries from **upsBatteryDetails.** Then store the minimum and maximum sold batteries name in string **List** and return it. **Note :** In list first add the minimum sold battery name and second one is maximum sold.
public Dictionary<string, long> SortByCount()	This method is used to display all the battery details available in the **upsBatteryDetails** in ascending order by sold count. The result should be **Dictionary**.

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

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

2. Call the methods accordingly and display the result.

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

Note:

- Keep the method 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.

Sample Input / Output:

1. Find battery details

2. Minimum and Maximum sold

3. Sort batteries by count

4. Exit

Enter your choice

1

Enter the sold count

55

Battery Details	**Sold Count**
Exide Powersafe Plus	55

1. Find battery details

2. Minimum and Maximum sold

3. Sort batteries by count

4. Exit

Enter your choice

2

Minimum Sold Battery is : Exide Powersafe Plus

Maximum Sold Battery is : APC Plastic RBC2

1. Find battery details

2. Minimum and Maximum sold

3. Sort batteries by count

4. Exit

Enter your choice

3

Battery Details	**Sold Count**
Exide Powersafe Plus	55
Z Powers	250
Zen	510
Exide Xpress	790
APC Plastic RBC2	800

1. Find battery details

2. Minimum and Maximum sold

3. Sort batteries by count

4. Exit

Enter your choice

4

10. **Global Technologies**

Global technologies , a renowned software company, is looking to reward their employees for their hard work and dedication by increasing their salaries based on their experience. You being their trusted software consultant with expertise in C#, develop a cutting-edge application that will assist in managing the salaries of the employees. This solution will ensure that the salary increase process is seamless, transparent, and in line with the company's guidelines.

Functionalities:

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

Class	**Properties**

Employee	int EmployeeId string EmployeeName double Salary int IncrementPercentage

In class **Service,** implement the below-given methods and also **Inherit** the class **Employee**.

Method	Description
public bool FindIncrementPercentage(int yearsOfExperience)	This method is used to calculate the incremented percentage of the salary of the employees. If the range of yearsOfExperience is between **1 and 5(both inclusive)** then set the Property IncrementPercentage as 15. If the range of yearsOfExperience is between **6 and 10(both inclusive)** then set the Property IncrementPercentage as 30. If the range of yearsOfExperience is between **11 and 15(both inclusive)** then set the Property IncrementPercentage as 45. If the yearsOfExperience is between 1 and 15(both inclusive) then return true, Otherwise return **false.**
public double CalculateIncrementedSalary()	This method is used to calculate the incremented salary of the employee and return the same. To calculate the incremented salary: **IncrementedSalary=Salary+((Salary*IncrementPercentage)/100)**

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

1. Get the **EmployeeId**,**EmployeeName**, **Salary and YearsOfExperience** value from the **user**.

2. Call the **FindIncrementPercentage** method and if it returns true then move on to next step, else display **Invalid Years of Experience**

3. Use the values in method **CalculateIncrementSalary** and display the values 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.

Sample Input/Output1:

Enter the employee id

101

Enter the employee name

James

Enter the salary

25000

Enter the number of years in experience

4

Incremented Salary - 28750

Sample Input/Output2:

Enter the employee id

101

Enter the employee name

James

Enter the salary

25000

Enter the number of years of experience

11.Asian Theatre

The Asian theatre is seeking to streamline their movie tracking system and improve their ticket booking process. As a seasoned software consultant with expertise in C#, you develop a comprehensive application that will meet their needs. This solution will allow the Con Theatre to efficiently manage all movie details, making it easier for them to organize their operations and provide a better experience for their customers.

Functionalities:

In class **Movie,** implement the below-given properties.

Class	Properties
Movie	string MovieTitle string SeatType string Dimension float Cost

In class **Service,** implement the below-given methods and also **Inherit** the class **Movie**.

Method	Description
public bool ValidateSeatType(string seatType)	This method is used to validate the seat type. In this method, pass the **seatType** as a parameter. If the seatType is **Gold** or **Diamond** or **Elite** then return **true,** Otherwise return **false.** **Note:** **SeatType** is **Case-sensitive**.
public bool ValidateDimension(string dimension)	This method is used to validate the dimension. In this method, pass the **dimension** as a parameter.

	If the dimension is **2D** or **3D** then return **true,** Otherwise return **false.** **Note:** **Dimension** is **Case-sensitive**.
public Movie FindCost()	This method is used to find the cost of the ticket. using **SeatType** and **Dimension** values present in the class **Movie,** find the cost, then store the result in the **Movie object** and return it. To find the **Cost**, please refer to the table given-below

Seat Type	Cost for 2D	Cost for 3D
Gold	190	240
Diamond	210	260
Elite	250	300

The output must contain:

1. Movie title

2. First letter of the seat type

3. Dimension

4. Cost

These must be separated by an underscore (_).

Output for ex:

Avatar_D_3D_260

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

1. Get the **MovieTitle**, **SeatType and Dimension** value from the **user**.

2. Call the **ValidateSeatType** method and if it returns true then move on to next step, else display **Invalid Seat Type**

3. Call the **ValidateDimension** method and if it returns true then move on to next step, else display **Invalid Dimension**

4. Use the values in method **FindCost** and display the values 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.

Sample Input/Output1:

Enter movie title

Avatar

Enter seat type

Gold

Enter dimension

3D

Avatar_G_3D_240

Sample Input/Output2:

Enter movie title

Avengers

Enter seat type

Platinum

Enter dimension

3D

Invalid Seat Type

Sample Input/Output3:

Enter movie title

The Independence Day

Enter seat type

Elite

Enter dimension

1D

Invalid Dimension

12. **Scenario:Boutique**

Olivia, the proprietor of a boutique clothing store catering to women, is seeking to expand her business. Her store, located in a fashionable area, offers a curated selection of unique, high-quality fashion items. To reach her next level of success, she plans to launch a new marketing campaign and establish an online platform featuring a delivery-charge calculator.
You being her software consultant, help her to do this by developing a C# application

Functionalities:

In class **Order,** implement the below-given properties.

Datatype	Property Name
string	CustomerCode
int	ProductId

string	ProductName
double	ProductPrice

In class **OrderDetails,** implement the below-given methods and also **Inherit** class **Order** .

Method	Description
public bool ValidateCustomerCode()	This method is used to validate the customer code. The customer code should have **five** characters. The first **two** characters should be in **uppercase alphabets** and the remaining **three** should be **numbers** If the above condition is passed then return **true**. Otherwise return **false** Also display **Invalid customer code** in Main method.
public Order CalculatePriceWithDeliveryCharge()	This method is used to calculate the price of the product with delivery charge. Set the Calculated price to the Property **ProductPrice** Then store the details such as **CustomerCode**, **ProductId**, **ProductName** and the updated **ProductPrice** with delivery charge in **Order** object and return it. Refer to the table given-below to calculate the price with delivery charge.

Formula :

Price With Delivery Charge = Product Price + Delivery Charge

Product Price	Delivery Charge
Less than 500	40 % of Product Price
Between 500 to 1000	10 % of Product Price
Above 1000	0 % of Product Price

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

1. Get the **CustomerCode** from the user.

2. Call the **ValidateCustomerCode** method and if it returns true, then get the **ProductId** , **ProductName** and **ProductPrice** value from the **user** and move on to step 3. If it returns **false**, then display **Invalid customer code**

3. Use the values in method **CalculatePriceWithDeliveryCharge** 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.

Sample Input 1:

Enter the customer code

AB123

Enter the product id

101

Enter the product name

Glamorous Gown

Enter the price

700

Sample Output 1:

Customer Code : AB123

Product Id : 101

Product Name : Glamorous Gown

Amount With Delivery Charge : 770

Sample Input 2:

Enter the customer code

AB123

Enter the product id

102

Enter the product name

Velvet Vixen Dress

Enter the price

2500

Sample Output 2:

Customer Code : AB123

Product Id : 102

Product Name : Velvet Vixen Dress

Amount With Delivery Charge : 2500

Sample Input 3:

Enter the customer code

SN12A

Sample Output 3:

Invalid customer code

13. **Scenario:**

As the sun shines brightly on this picturesque spring day, Maria, the proprietor of The Plant Haven, a thriving plant store in the heart of the town, busies herself with curating a diverse array of botanical offerings for her valued customers. From colourful annuals and hardy perennials to fragrant herbs and nutritious vegetables, she has something for everyone. To sweeten the deal, she even offers generous discounts to her patrons. As the day progresses, Maria is delighted by the steady stream of customers who come through her doors. Recognizing the need to streamline her operations, she decides to invest in software that can accurately calculate the cost of her plants after discounts have been applied.
You being her software consultant, help her by developing a C# application

Functionalities:

In class **Plant,** implement the below-given properties.

Datatype	Property Name
string	PlantName
int	NoOfSapling
string	Category
int	PricePerSapling

In class **PlantUtility,** implement the below-given methods and also **Inherit** class **Plant**.

Method	Description
public Plant ExtractDetails(string plantDetails)	This method is used to extract plant details from a **colon(:) - separated** input **string** and assign the values to **Plant** class **object** and return that**.** In this method, pass the plant details as an argument based on the below-given string format. **<PlantName:NoOfSapling:Category:PricePerSapling>**
public double CalculateCost()	This method is used to calculate the cost of the plant after the discount and return it as a **double** datatype.

	Total Amount = **NoOfSapling * PricePerSapling** Refer to the below procedure to calculate cost after the discount.

Formula :

Cost after the discount = Total Amount - Discount

Total Amount	Discount
> 500 to <=1000	10 % of Total Amount
Above 1000	20 % of Total Amount
<=500	No discount

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

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

2. Call the **ExtractDetails** method by using the input values.

3. Call the **CalculateCost** method and finally display all the details as per the output given.

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.

Sample Input 1:

Enter the plant details

Sunflower:10:flowering:70

Sample Output 1:

Plant name is : Sunflower

No of sapling is : 10

Category : flowering

Price per sapling : 70

Total cost is : 630

Sample Input 2:

Enter the plant details

Lavender:5:herb:30

Sample Output 2:

Plant name is : Lavender

No of sapling is : 5

Category : herb

Price per sapling : 30

Total cost is : 150

14. **Konark Health Care**

A large multispecialty hospital, Konark Health Care, has recently started administering COVID-19 vaccines to its patients. The hospital management needs an application that can help them analyse and manipulate the booking details for the vaccine doses. The application should be able to track the number of doses available for each vaccine type and the number of doses that have been administered.

As their software consultant, you can help them by developing a C# application.

Functionalities:

In class **Vaccine,** implement the below-given properties.

Data Type	**Properties**
string	BookingId
string	Name
string	VaccineType

string	DoseNumber
string	BookingDate

In class **Program,** implement the below-given method.

public static List<Vaccine> VaccineList -In the code template, it is already provided.

implement the features listed below.

Method	Description
public void AddVaccineDetails (string [ ] vaccineDetails)	This method is used to add the vaccine booking details to the **VaccineList**. This method should separate the values in each string by a **comma** from the **string array** passed as an argument and convert it into a Vaccine object and each **Vaccine** object should be stored in a **VaccineList**.
public List<Vaccine> ViewBookingDetailsByDoseNumber(string doseNumber)	This method is used to find the booking details based on the **doseNumber** passed as an argument. If the dose number is available in the **VaccineList** then store the booking details as **List** of Vaccine and return it. If the dose number is not available in the **VaccineList,** then return an empty **List** and print "**Does number not found**" in Main method.
public List<Vaccine> ViewBookingDetailsByVaccineType(string vaccineType)	This method is used to find the booking details based on the **vaccineType** passed as an argument. If the vaccine type is available in the **VaccineList**, then store the booking details as **List** of Vaccine and return it. If the vaccine type is not available in the **VaccineList,** then return an empty **List** and print "**Vaccine type not found**" in Main method.

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

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

2. Call the methods accordingly and display the result.

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

Note:

- Keep the method 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.

Sample Input / Output:

1. Add Vaccine Details

2. View Details By Dose Number

3. View Details By Vaccine Type

4. Exit

Enter the choice

1

Enter the number of entries

3

M001, Raj, COVIDSHIELD, DOSE 1, 25/12/2022

M002, Alex, COVAXIN, DOSE 1, 25/12/2022

M003, Mohammed, COVIDSHIELD, DOSE 2, 25/12/2022

1. Add Vaccine Details

2. View Details By Dose Number

3. View Details By Vaccine Type

4. Exit

Enter the choice

2

Enter the dose number

DOSE 1

M001 Raj COVIDSHIELD DOSE 1 25/12/2022

M002 Alex COVAXIN DOSE 1 25/12/2022

1. Add Vaccine Details

2. View Details By Dose Number

3. View Details By Vaccine Type

4. Exit

Enter the choice

2

Enter the dose number

DOSE 4

Does number not found

1. Add Vaccine Details

2. View Details By Dose Number

3. View Details By Vaccine Type

4. Exit

Enter the choice

3

Enter the vaccine type

COVIDSHIELD

M001 Raj COVIDSHIELD DOSE 1 25/12/2022

M003 Mohammed COVIDSHIELD DOSE 1 25/12/2022

1. Add Vaccine Details

2. View Details By Dose Number

3. View Details By Vaccine Type

4. Exit

Enter the choice

4

`15. **Pearl Drink app**

Pearl Drinks is a medium-sized beverage company that specializes in producing a variety of healthy and refreshing drinks. The company has recently decided to launch a new line of plant-based protein drinks to appeal to a wider audience of health-conscious consumers. The company will also launch a marketing campaign to promote the new line, targeting fitness enthusiasts and health-conscious consumers through social media and partnerships with fitness studios and gyms. They need software to manage their beverage details.

As their software consultant, you can help them by developing a C# application.

Functionalities:

In class **Program,** implement the below-given method.

public static Dictionary<String, int> beverageDetails -In the code template, it is already provided.

Implement the features listed below.

Method	Description

public Dictionary<String, int> SearchBeverage(String beverageName)	This method is used to find the beverage details by beverage name. If the beverage is available in the **beverageDetails** dictionary**,** it should return the **beverage name** and **price** as **Dictionary**. If the beverage is not available in the **beverageDetails** dictionary**,** then it should return an empty **Dictionary** and print **"Beverage Not Found"** in **Main** method.
public Dictionary<String, int> UpdateBeveragePrice(string beverageName, int price)	This method is used to update the **price** of the beverage by **beverageName**. If the beverage is available in the **beverageDetails** dictionary**,** it should return the updated details as **Dictionary**. If the beverage is not available in the **beverageDetails** dictionary**,** then should return an empty **Dictionary** and print **"Beverage Not Found"** in **Main** method.
public Dictionary<String, int> SortByBeverageName()	This method is used to display all the beverages details available in the **beverageDetails** dictionary in ascending order by beverage name. The result should be stored as **Dictionary**.

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

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

2. Call the methods accordingly and display the result.

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

Note:

- Keep the method 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.

Sample Input / Output:

1. Search by beverage name

2. Update beverage price

3. Sort beverages by name

4. Exit

Enter your choice

1

Enter the beverage name

Pepsi

Pepsi 230

1. Search by beverage name

2. Update beverage price

3. Sort beverages by name

4. Exit

Enter your choice

2

Enter the beverage name

Fanta

Enter the beverage price

270

Fanta 270

1. Search by beverage name

2. Update beverage price

3. Sort beverages by name

4. Exit

Enter your choice

2

Enter the beverage name

Lime

Enter the beverage price

500

Beverage Not Found

1. Search by beverage name

2. Update beverage price

3. Sort beverages by name

4. Exit

Enter your choice

3

Citrus Soda 100

Cola 250

Fanta 270

Lemonade 150

Pepsi 230

1. Search by stock name

2. Update stock price

3. Sort stocks by price

4. Exit

Enter your choice

4

16. **Corus Concert app**

The Corus Concert is a music concert production and promotion company that works with a diverse range of musicians and artists, from local up-and-coming acts to internationally renowned performers. They handle all aspects of concert production, including venue booking, logistics, marketing, and ticket sales. As such, they require software to manage and calculate all aspects of their operations.

You being their software developer, help them to do the task by developing a C# application.

Functionalities:

In class **Concert,** implement the below-given properties.

Datatype	Property Name
DateTime	Date
string	SeatingType
int	VisitorsCount

In class **ConcertDetails,** implement the below-given methods and also **Inherit** the class **Concert**.

Method	Description
public bool ValidateDay()	This method is used to extract the day from the date and validate the day.

	If the day is **Saturday** or **Sunday**, then return **true**. Otherwise, return **false**. Note : **Saturday** and **Sunday** is **Case-sensitive**. Date Format : **MM/DD/YYYY**
public double TicketPriceCalculation()	This method is used to calculate the price of the ticket based on seating type class and return that price as a **double** datatype. Calculate the Ticket price based on the below-given formula.

Formula :

Ticket Price = (Visitors Count * Price Per Visitor) + Extra Charge

Seating Type Class	Price Per Visitor	Extra Charge
First	2000	20 % of Price Per Visitor
Second	1000	10% of Price Per Visitor
Normal	500	No extra charge

Note:

Seating Type Class is **Case-sensitive**.

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

1. Get the date from the **user**.

2. Call the **ValidateDay** method. If it returns true, then get the seating type and visitors count from the user and move on to step 3. If this method returns false, then display **"Ticket is not available".**

3. Call the **TicketPriceCalculation** method and display the price details 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.

Sample Input 1:

Enter the date

10/10/2020

Enter the seating type

First

Enter the visitors count

5

Sample Output 1:

The ticket cost is 10400

Sample Input 2:

Enter the date

01/01/2023

Enter the seating type

Normal

Enter the visitors count

1

Sample Output 2:

The ticket cost is 500

Sample Input 3:

Enter the date

01/03/2022

Sample Output 3:

Ticket is not available

17. Financial App

Yes Bank is a financial institution that accepts deposits from the public and creates credit. They decides to provide a cashback offer for each debit card withdrawal in order to increase card usage. They need software to manage credit details and provide the cashback.

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

Functionalities:

In class **Card,** implement the below-given properties.

Data Type	Property Name
long	CardNumber
long	BalanceAmount

In class **Service,** implement the below-given methods and also **Inherit** the class **Card**.

Method	Description
public bool ValidateCardNumber()	This method is used to validate the card number. If the card number contains **16** digits, then return **true**. Otherwise, return **false.**
public double[ ] Withdraw(long withdrawnAmount)	This method takes the amount to be withdrawn as argument. This method should check the balance amount in the class **Card** and detect the withdrawn amount from the

	balance amount and calculate the cashback and return the **balance amount** and **cashback** as an array. If the withdrawn amount must be less than or equal to balance amount. Otherwise return an **empty array**. If this method returns an empty array, then display "**Insufficient Balance**" in Main method.

Formula :

Balance Amount = Balance Amount - withdrawn Amount

Cashback = withdrawn Amount * cashback %

Calculation for Cashback:

Withdrawn Amount	**Cashback Percentage**
>=100 and < =500	10%
> 500 and <= 1000	15%
>1000	20%
<100	0%

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

1. Get the **CardNumber** from the user.

2. Call the **ValidateCardNumber** method, If it returns true then get the **Balance Amount** and **Withdrawn Amount** value from the **user** and move on to step 3, If it returns false then display **Invalid Card Number**

3. Use the values in method **Withdraw** 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.

Sample Input 1:

Enter the card number

1234567890123456

Enter the card limit

15000

Enter the amount to be withdrawn

500

Sample Output 1:

Available balance is: 14500

Cashback is: 50

Sample Input 2:

Enter the card number

1234567890123456

Enter the card limit

15000

Enter the amount to be withdrawn

18500

Sample Output 2:

Insufficient Balance

Sample Input 3:

Enter the card number

12453

Sample Output 3:

Invalid Card Number

18. Fashion Store

A Vin fashion store's vision might involve providing stylish and high-quality clothing and accessories to customers and helping them feel confident and fashionable. The store may strive to be a destination for fashion-forward individuals who appreciate the latest trends and unique, original styles. To help a fashion store maintain its products, you could develop a C# application that allows the store to track and manage its inventory, sales, and customer information.

Functionalities:

In class **Product**, implement the below-given properties.

Data Type	Properties
int	Id
string	Name
double	Price

In class **Program,** implement the below-given method.

public static Dictionary<int, Product> products -In the code template, it is already provided.

implement the features listed below.

Method	Description
public Dictionary<int, Product> SearchProduct(String productName)	This method is used to find the product details by **productName**. If the product is available in the **products** dictionary**,** it should return the details such as Id, Name, and Price as **Dictionary**. If the product is not available in the **products** dictionary**,** then it should return an

	empty **Dictionary and** print **"Product Not Found"** in **Main** method.
public Dictionary<int, Product> UpdateProductPrice(string productName, int price)	This method is used to update the price of the product by product name. If the product is available in the **products** dictionary**,** it should update the product **price** by **productName** and return the updated details such as Id, Name, and Price as **Dictionary**. If the product is not available in the **products** dictionary**,** then it should return an empty **Dictionary and** print **"Product Not Found"** in **Main** method.
public Dictionary<int, Product> SortByProductName()	This method is used to display all the products available in the **products** dictionary in ascending order by **ProductName**. The return result should be **Dictionary**.

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

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

2. Call the methods accordingly and display the result.

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

Note:

- Keep the method 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.

Sample Input / Output:

1. Search product by name

2. Update product price

3. Sort product by name

4. Exit

Enter your choice

1

Enter a product name to search:

Shirt

Product ID: 1 Name: Shirt Price: 450

1. Search product by name

2. Update product price

3. Sort product by name

4. Exit

Enter your choice

2

Enter a product name to update:

Pant

Enter a new price for the product:

1000

Product ID: 2 Name: Pant Price: 1000

1. Search product by name

2. Update product price

3. Sort product by name

4. Exit

Enter your choice

2

Enter a product name to update:

Watch

Enter a new price for the product:

500

Product Not Found

1. Search product by name

2. Update product price

3. Sort product by name

4. Exit

Enter your choice

3

Product ID: 2 Name: Pant Price: 1000

Product ID: 1 Name: Shirt Price: 450

Product ID: 3 Name: Shoe Price: 250

1. Search product by name

2. Update product price

3. Sort product by name

4. Exit

Enter your choice

4 Atomsphere data

The Regional Meteorological Center is responsible for collecting, analyzing and storing Atmosphere data for the entire region throughout the year. To ensure that the data is easily accessible and usable, RMC wants to create a Atmosphere analysis and manipulation application. This application will be used by meteorologists, researchers, and other staff members to easily retrieve and analyze data based on location and date.

You being their software consultant, help them by developing a C# application.

Functionalities:

In class **Atmosphere**, implement the below-given properties.

Data Type	**Properties**
string	Location
string	Date
int	Temperature
string	Status

In class **Program,** implement the below-given method.

public static List<Atmosphere> AtmosphereList -In the code template, it is already provided.

implement the features listed below.

Method	**Description**
public void AddAtmosphereDetails (string[ ] AtmosphereDetails)	This method is used to add the Atmosphere details into the **AtmosphereList.** This method should separate the values in each string by a **comma** from the input array and assign those values to the **Atmosphere** object, then add each object to the **AtmosphereList.**

public List<Atmosphere> ViewDetailsByLocation(string location)	This method is used to find the Atmosphere details based on the **location** passed as an argument. If the **location** is found in the **AtmosphereList**, then add those Atmosphere details to the List and return that **List**. else, return an **empty list.**
public List<Atmosphere> ViewDetailsByDate(string date)	This method is used to find the Atmosphere details based on the **date** passed as an argument. If the **date** is found in the **AtmosphereList**, then add those Atmosphere details to the List and return that **List**. else, return an **empty list.**

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

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

2. Call the methods accordingly and display the result.

-- If the **ViewDetailsByLocation** method returns an empty list, then display **"Location is not found".**

-- If the **ViewDetailsByDate** method returns an empty list, then display **"Date is not found".**

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

Note:

- Keep the method 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.

Sample Input and Output :

1. Add Atmosphere Details

2. View Details By Location

3. View Details By Date

4. Exit

Enter the choice

1

Enter the number of entries

3

Miami,25/01/2023,20,Partly Cloud

Miami,26/01/2023,33,Mostly Sunny

Berth,25/01/2023,25,Sunny

1. Add Atmosphere Details

2. View Details By Location

3. View Details By Date

4. Exit

Enter the choice

2

Enter the location

Miami

Location	Date	Temperature	Status
Miami	25/01/2023	20	Partly Cloud
Miami	26/01/2023	33	Mostly Sunny

1. Add Atmosphere Details

2. View Details By Location

3. View Details By Date

4. Exit

Enter the choice

2

Enter the location

Sydney

Location is not found

1. Add Atmosphere Details

2. View Details By Location

3. View Details By Date

4. Exit

Enter the choice

3

Enter the date

25/01/2023

Location	Date	Temperature	Status
Miami	25/01/2023	20	Partly Cloud
Berth	25/01/2023	25	Sunny

1. Add Atmosphere Details

2. View Details By Location

3. View Details By Date

4. Exit

Enter the choice

3

Enter the date

10/01/2000

Date is not found

1. Add Atmosphere Details

2. View Details By Location

3. View Details By Date

4. Exit

Enter the choice

4

Thank you.

19. Hill Height

Hamels is an experienced traveller and an avoid hill. He has decided that his next adventure will be climbing a hill and he wants to choose one that is particularly tall. He has researched several hills and narrowed down his options to a few that he is interested in climbing.

However, he wants to make sure that he chooses the tallest one out of the options he has. He wants software that provides information on the height of different hills around the world.

You being his software consultant help him by developing a C# application.

Functionalities:

In class **Program,** implement the below-given method.

public static Dictionary<string, int> HillDetails -In the code template, it is already provided.

implement the features listed below.

Method	Description
public void AddHillDetails (string[ ] hill)	This method is used to add the hill details into the **HillDetails** dictionary. This method should separate the values in each string by a **colon(:)** from the input array and store them in a **Dictionary**.
public int FindHillHeight (String hillName)	This method is used to find the height of the hill based on the **hillName** passed as an argument. If the hill name is found in the Dictionary, then return that hill **height**. else, return **-1** and print "**No hills are available**" in the Main method.
public List <String> FindTheHighestHills()	This method is used to find the highest height hill from the **HillDetails** dictionary, then store the hill name as a **list** and return it. **Note:** If the highest height has more than one hill, then consider that all are the highest height hills, and they need to be added to the list.

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

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

2. Call the methods accordingly and display the result.

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

Note:

- Keep the method 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.

Sample Input and Output :

1. Add Hill Details

2. View Hill Height

3. View Hills With Highest Height

4. Exit

Enter the choice

1

Enter the number of entries

5

Mount Elbrus:7970

Aconcagua:7827

Manaslu:4510

Mount Vinson:7970

K2:6120

1. Add Hill Details

2. View Hill Height

3. View Hills With Highest Height

4. Exit

Enter the choice

2

Enter the hill name needs to be searched

K34

No hills are available

1. Add Hill Details

2. View Hill Height

3. View Hills With Highest Height

4. Exit

Enter the choice

2

Enter the hill name needs to be searched

Aconcagua

Height is : 7827

1. Add Hill Details

2. View Hill Height

3. View Hills With Highest Height

4. Exit

Enter the choice

3

Hill names with the highest height are:

Mount Elbrus

Mount Vinson

1. Add Hill Details

2. View Hill Height

3. View Hills With Highest Height

4. Exit

Enter the choice

4