Program- **VISIONMOBILES**

```
using System;

using System.Collections.Generic;

 namespace VisionMobiles

{
public class Program

{
public static SortedDictionary<string, long> mobileDetails = new SortedDictionary<string,
long>()
 {

   {"Nokia", 55),

   {"Samsung", 250),

    {"Sony", 510},

    {"Oneplus", 790},

    {"Redmi", 800}
};

public static void Main(string[] args)
{

int choice = 0;

while (choice != 4)

{

Console.WriteLine("1. Find mobile details");

Console.WriteLine("2. Minimum and Maximum sold");

Console.WriteLine("3. Sort mobiles by count");

Console.WriteLine("4. Exit");

Console.WriteLine("Enter your choice");
int.TryParse(Console.ReadLine(), out choice);

switch (choice)
{
 case 1:
   Console.WriteLine("Enter the sold count");

    long soldCount = long.Parse(Console.ReadLine());

    SortedDictionary<string, long> mobileDetailsBySoldCount =
```

```
FindMobileDetails(soldCount);

if (mobileDetailsBySoldCount.Count > 0)

{

foreach (var mobileDetail in mobileDetailsBySoldCount)

{
Console.WriteLine($"{mobileDetail.Key} {mobileDetail.Value}");
}


}
Else
{
 Console.WriteLine("Invalid sold count");

}

break;

case 2:

List<string> minAndMaxSoldMobiles = FindMinandMaxSoldMobiles();

Console.WriteLine($"Minimum Sold Mobile is: {minAndMaxSoldMobiles[@]}");
Console.WriteLine($"Maximum Sold Mobile is: (minAndMaxSoldMobiles[1]}");

break;

case 3:

Dictionary<string, long> sortedMobileDetails=SortByCount();

foreach (var mobileDetail in sortedMobileDetails)
{
Console.WriteLine($"{mobileDetail.Key} {mobileDetail.Value}");
}
break;

case 4:

Console.WriteLine("Thank You");

break;

default:

Console.WriteLine("Invalid choice");

break;
}
}
}
```

```
public static SortedDictionary<string, long> FindMobileDetails(long soldCount) {


SortedDictionary<string, long> mobileDetailsBySoldCount = new SortedDictionary<string,
long>();


foreach (var mobileDetail in mobileDetails)

if (mobileDetail.Value == soldCount)

{

}

}

mobileDetailsBySoldCount.Add(mobileDetail.Key, mobileDetail.Value); }

return mobileDetailsBySoldCount;


public static List<string> FindMinandMaxSoldMobiles()

{

long minSoldCount long.MaxValue;

long maxSoldCount long. MinValue;

string minSoldMobile = "";

string maxSoldMobile = "";

foreach (var mobileDetail in mobileDetails)

{ if (mobileDetail.Value < minSoldCount)

{ minSoldCount = mobileDetail.Value; minSoldMobile = mobileDetail.Key;


if (mobileDetail.Value> maxSoldCount)

{ } maxSoldCount maxSoldMobile mobileDetail.Value; mobileDetail.Key;

}


}
```

```
return new List<string>() { minSoldMobile, maxSoldMobile };

public static Dictionary<string, long SortByCount()

{

List<KeyValuePair<string, long>> sortedMobileDetails new List<KeyValuePair<string,
long>>(mobileDetails); sortedMobileDetails.Sort((x, y) => x.Value.CompareTo(y.Value));

Dictionary<string, long> sortedMobiles = new Dictionary string, long();
Dictionary<string, long sortedMobiles = new Dictionary<string, long>();

foreach (var mobileDetail in sortedMobileDetails)

{ sortedMobiles.Add(mobileDetail.Key, mobileDetail.Value);

}
return sortedMobiles;

}

}
```

Program- **TakeOutRestaurant**

Program.cs

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;


namespace TakeOutRestaurant //DO NOT change the namespace name

{
public class Program

public static void Main(string[] args)

{

Console.WriteLine("Enter the food type");

string food Type =Console.ReadLine();
```

```
Console.WriteLine("Enter the quantity");

int quantity =Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Enter the price per piece");

int pricePerPiece = Convert.ToInt32(Console.ReadLine());

Billing billing = new Billing()
{
  FoodType = foodType,
Quality = quality,
PricePerPiece = pricePerPiece
};

if (billing. ValidateFoodType(billing. FoodType)) {

FoodDetails billDetails billing.GenerateBill();

Console.WriteLine("FoodType");

Console.WriteLine("Quantity");

Console.WriteLine("PricePerPiece"); Console.WriteLine("TotalPrice");

Console.WriteLine("Discount");

Console.WriteLine(billDetails.FoodType);

Console.WriteLine(billDetails.Quantity).;

Console.WriteLine(billDetails.PricePerPiece); Console.WriteLine(billDetails. TotalPrice);


Console.WriteLine(billDetails.Discount);

}

}
}
```

FoodDetails.cs

```
using System;


using System.Collections.Generic;
```

Evaluate

```
using System.Linq;

using System.Text;

 using System.Threading.Tasks;


namespace TakeOutRestaurant //DO NOT change the namespace name
{


public class FoodDetails

{


public string FoodType { get; set; }


public int Quantity { get; set; }

public int PricePerPiece { get; set; }


public double TotalPrice { get; set; }

public double Discount { get; set; }
}

}
```

Billing.cs

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;


namespace TakeOutRestaurant //DO NOT change the namespace name
```

```
{

public class Billing FoodDetails
{
public bool Validate FoodType(string foodType)

{

if (foodType == "Samosa" || foodType == "Spring Roll" || foodType == "Empanada");
 return true;

Console.WriteLine("Invalid food type");

return false;
}

public FoodDetails GenerateBill()

{

TotalPrice = Quantity* PricePerpiece;


if (TotalPrice >= 100 && TotalPrice 500)
    Discount = TotalPrice 0.10;

else if (TotalPrice > 500 && TotalPrice <= 1000)
    Discount = TotalPrice 0.15;

else if (TotalPrice 1000)
    Discount =TotalPrice 0.20;
else
Discount =0;

FoodDetails foodDetails = new FoodDetails()

{

FoodType = FoodType,

Quantity = Quantity,
PricePerPiece = PricePerPiece,

TotalPrice = TotalPrice,

Discount = Discount

};

return foodDetails;
}
}
```

Bike

Program.cs:

csharp

```
using System;

class Program
{
    static void Main()
    {
        Console.WriteLine("Enter the bike number");
        string bikeNumber = Console.ReadLine();

        BikeUtility bikeUtility = new BikeUtility();

        if (bikeUtility.ValidateBikeNumber(bikeNumber))
        {
            Console.WriteLine("Enter the engine capacity");
            int engineCapacity = int.Parse(Console.ReadLine());

            Console.WriteLine("Enter the year");
            int year = int.Parse(Console.ReadLine());

            Console.WriteLine("Enter the cost of the bike");
            double cost = double.Parse(Console.ReadLine());

            Bike bike = bikeUtility.CalculateInsurance(bikeNumber, engineCapacity, year, cost);

            Console.WriteLine("BikeNumber\tEngineCapacity\tYear\tCost\tInsurance");

Console.WriteLine($"{bike.BikeNumber}\t\t{bike.EngineCapacity}\t\t{bike.Year}\t{bike.Cost}\t{
bike.Insurance}");
        }
        else
        {
            Console.WriteLine("Invalid bike number");
        }
    }
}
```

Bike.cs:

csharp

```
public class Bike
{
    public string BikeNumber { get; set; }
    public int EngineCapacity { get; set; }
    public int Year { get; set; }
    public double Cost { get; set; }
    public double Insurance { get; set; }
}
```

BikeUtility.cs:

csharp

```
public class BikeUtility : Bike
{
    public bool ValidateBikeNumber(string bikeNumber)
    {
        if (bikeNumber.Length == 5 && char.IsLetter(bikeNumber[0]) &&
char.IsLetter(bikeNumber[1]) && char.IsDigit(bikeNumber[2]) && char.IsDigit(bikeNumber[3])
&& char.IsDigit(bikeNumber[4]))
        {
            return true;
        }
        else
        {
            Console.WriteLine("Invalid bike number");
            return false;
        }
    }

    public Bike CalculateInsurance(string bikeNumber, int engineCapacity, int year, double
cost)
    {
        Bike bike = new Bike
        {
            BikeNumber = bikeNumber,
            EngineCapacity = engineCapacity,
            Year = year,
            Cost = cost
        };

        if (engineCapacity <= 200)
        {
            if (year <= 2000)
            {
                bike.Insurance = 0.01 * cost;
            }
            else
            {
                bike.Insurance = 0.02 * cost;
            }
        }
        else
        {
            if (year <= 2000)
            {
                bike.Insurance = 0.03 * cost;
            }
            else
            {
                bike.Insurance = 0.04 * cost;
            }
        }
```

```
        return bike;
    }
}
```

GET HOLIDAY

:

```
csharp
using System;
using System.Collections.Generic;
using System.Linq;

public class Program
{
    public static Dictionary<string, float> hotelDetails = new Dictionary<string, float>
    {
        {"The Hay Adams", 3.0f},
        {"Montage Kapalua Bay", 4.0f},
        {"Jungle Resort", 4.5f},
        {"Mandarin Oriental", 5.0f},
        {"The Greenwich Hotel", 5.0f}
    };

    public static void Main()
    {
        int choice;

        do
        {
            Console.WriteLine("1. Search by hotel name");
            Console.WriteLine("2. Update hotel rating");
            Console.WriteLine("3. Sort hotels by name");
            Console.WriteLine("4. Exit");
            Console.WriteLine("Enter your choice");

            if (int.TryParse(Console.ReadLine(), out choice))
            {
                switch (choice)
                {
                    case 1:
                        Console.WriteLine("Enter the hotel name");
                        string searchHotelName = Console.ReadLine();
                        var searchResult = SearchHotel(searchHotelName);
                        if (searchResult.Count > 0)
                        {
                            Console.WriteLine($"{searchResult.Keys.First()}
{searchResult.Values.First()}");
                        }
                        else
                        {
```

```
                    Console.WriteLine("Hotel Not Found");
                }
                break;

            case 2:
                Console.WriteLine("Enter the hotel name");
                string updateHotelName = Console.ReadLine();
                Console.WriteLine("Enter the rating");
                float updatedRating;
                if (float.TryParse(Console.ReadLine(), out updatedRating))
                {
                    var updateResult = UpdateHotelRating(updateHotelName, updatedRating);
                    if (updateResult.Count > 0)
                    {
                        Console.WriteLine($"{updateResult.Keys.First()}
{updateResult.Values.First()}");
                    }
                    else
                    {
                        Console.WriteLine("Hotel Not Found");
                    }
                }
                else
                {
                    Console.WriteLine("Invalid rating input");
                }
                break;

            case 3:
                var sortedHotels = SortByHotelName();
                foreach (var hotel in sortedHotels)
                {
                    Console.WriteLine($"{hotel.Key} {hotel.Value}");
                }
                break;

            case 4:
                Console.WriteLine("Thank You");
                break;

            default:
                Console.WriteLine("Invalid choice");
                break;
        }
    }
    else
    {
        Console.WriteLine("Invalid input");
    }

} while (choice != 4);
}

public static Dictionary<string, float> SearchHotel(string hotelName)
{
```

```
        if (hotelDetails.ContainsKey(hotelName))
        {
            return new Dictionary<string, float> { { hotelName, hotelDetails[hotelName] } };
        }
        else
        {
            return new Dictionary<string, float>();
        }
    }

    public static Dictionary<string, float> UpdateHotelRating(string hotelName, float rating)
    {
        if (hotelDetails.ContainsKey(hotelName))
        {
            hotelDetails[hotelName] = rating;
            return new Dictionary<string, float> { { hotelName, rating } };
        }
        else
        {
            return new Dictionary<string, float>();
        }
    }

    public static Dictionary<string, float> SortByHotelName()
    {
        var sortedHotels = hotelDetails.OrderBy(h => h.Key).ToDictionary(h => h.Key, h =>
h.Value);
        return sortedHotels;
    }
}
```

.