1)Get Holiday

2)Income tax

3)vision mobile

4)alia cabs

5)crazy franky cakes

6)plant heaven

7)carslon bike insurance

8)madoff's minions

9)win cineams

10)revv bikes

11)secret trade

12)elegant jewllers

13)destuche bank

---

1)Get Holiday

```
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");
   return;
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;
    }
}
```

---

2)Income tax

```
using System;
using System.Text.RegularExpressions;
public class Employee
{
    public string EmployeeId{get; set;}
    public double Salary{get; set;}

}
public class EmployeeUtility : Employee
{
    public bool ValidateEmployeeId()
    {
        if(EmployeeId.Length == 4)
        {
            string pattern = @"^[A-Z]\d{3}$";
            if(Regex.IsMatch(EmployeeId, pattern))
            {
                return true;
            }
```

```
        else
        {
            return false;
        }

    }
    else
    {
    return false;
    }
}



public double calculateTaxAmount()
{
    double tax = 0.0;
    if(Salary <= 20000)
    {
        tax = 0;
    }
    else if(Salary > 20000 && Salary <= 50000)
    {
        tax = ((Salary -20000)*0.10);
    }
    else if(Salary > 50000 && Salary < 100000)
    {
        tax = ((30000)*0.10) + ((Salary-50000)*0.20);
    }
    else
    {
```

```
        tax = ((30000)*0.10) + ((50000)*0.20) + ((Salary-100000)*0.30);
      }
   return tax;
   }
}
public class Program
{
   public static void Main(string[] args)
   {
      EmployeeUtility eu = new EmployeeUtility();

      Console.WriteLine("Enter the Employee Id");
      eu.EmployeeId = Console.ReadLine();

      if(eu.ValidateEmployeeId())
      {
         Console.WriteLine("Enter the Salary");
         eu.Salary = Convert.ToDouble(Console.ReadLine());
         Console.WriteLine("Total Tax amount is :"+eu.calculateTaxAmount());
      }
      else
      {
         Console.WriteLine("Invalid Employee Id");
      }



   }
}
```

---

3)vision mobile

```
using System;
using System.Collections.Generic;
using System.Linq;
public class Program
{
   public static void Main()
   {
      Program pr=new Program();
      while(true)
      {
         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 choice=int.Parse(Console.ReadLine());
         switch(choice)
         {
            case 1:
               Console.WriteLine("Enter the sold count");
               int sc=int.Parse(Console.ReadLine());
               SortedDictionary<string,long> k=pr.FindMobileDetails(sc);
               foreach(var item in k)
               {
                  Console.WriteLine(item.Key+" "+item.Value);
```

```
            }
            break;
        case 2:
            List<string> minandmax=pr.FindMinandMaxSoldMobiles();
            Console.WriteLine("Minimum Sold Mobile is : "+minandmax[0]);
            Console.WriteLine("Maximum Sold Mobile is : "+minandmax[1]);
            break;
        case 3:
            Dictionary<string,long> sor=pr.SortByCount();
            foreach(var item in sor)
            {
               Console.WriteLine(item.Key+" "+item.Value);
            }
            break;
        case 4:
            Console.WriteLine("Thank you");
            return ;


      }
   }

}
public static SortedDictionary<string,long> mobileDetails = new SortedDictionary<string,long>()
{
   {"Nokia",55},
   {"Samsung",250},
   {"Sony",510},
   {"Oneplus",790},
   {"Redmi",800}
};
```

```
public SortedDictionary<string,long> FindMobileDetails(long soldCount)
{
   SortedDictionary<string,long> sd=new SortedDictionary<string,long>();
   foreach(var item in mobileDetails)
   {
      if(item.Value==soldCount)
      {
         sd.Add(item.Key,item.Value);
      }
   }
   return sd;
}
public List<string> FindMinandMaxSoldMobiles()
{
   List<string> name=new List<string>();
   long res=long.MaxValue;
   long res1=long.MinValue;
   foreach(var item in mobileDetails)
   {
      res=Math.Min(item.Value,res);
      res1=Math.Max(item.Value,res1);
   }
   foreach(var item in mobileDetails)
   {
      if(res==item.Value)
      {
         name.Add(item.Key);
      }

   }
   foreach(var item in mobileDetails)
```

```
        {
            if(res1==item.Value)
            {
                name.Add(item.Key);
            }

        }
        return name;
    }
    public Dictionary<string,long> SortByCount()
    {
      return mobileDetails.OrderBy(c=>c.Value).ToDictionary(a=>a.Key,a=>a.Value);
    }
}
```

---

4)alia cabs

```
using System;
public class Cab
{
    public string BookingID{get;set;}
    public string CabType{get;set;}
    public double Distance{get;set;}
    public double Fare{get;set;}

}
```

```
public class CabDetails:Cab
{
   public bool ValidateBookingID()
   {
      if(BookingID.StartsWith("AC@")&&BookingID.Length==6 &&
int.TryParse(BookingID.Substring(3),out int num))
      {
         return true;
      }
      return false;
   }
   public Cab CalculateFareAmount()
   {
      double pricePerkm=0;
     if(CabType=="Hatchback")
     {
        pricePerkm=10;
     }
     else if(CabType=="Sedan")
     {
        pricePerkm=20;
     }
     else
     {
        pricePerkm=30;
     }
     Fare=Distance*pricePerkm;
     return this;
   }
}
```

```
public class Program
{
   public static void Main(string[] args)
   {
      CabDetails cab = new CabDetails();
      Console.WriteLine("Enter the booking ID:");
      cab.BookingID = Console.ReadLine();
      if (cab.ValidateBookingID())
      {
         Console.WriteLine("Enter the cab Type:");
         cab.CabType = Console.ReadLine();
         Console.WriteLine("Enter the distance in km:");
         cab.Distance = double.Parse(Console.ReadLine());
         cab.CalculateFareAmount();
         Console.WriteLine("Booking Id: "+cab.BookingID);
         Console.WriteLine("CabType : "+cab.CabType);
         Console.WriteLine("Distance : "+cab.Distance);
         Console.WriteLine("Fare : "+cab.Fare);
      }
      else
      {
         Console.WriteLine("Invalid booking ID.");
      }
   }
}
```

---

5)crazy franky cakes

```
using System;
class Cake
{
   public string CakeType{get;set;}
   public string Flavour{get;set;}
   public int Quantity{get;set;}
   public int PricePerKg{get;set;}
   public double TotalPrice{get;set;}

   public Cake()
   {

   }
}
class CakeUtility:Cake
{
   public bool ValidateCakeType()
   {
      if(CakeType=="Butter"||CakeType=="Sponge"||CakeType=="Chiffon")
      {
         return true;
      }
      else
      {
         Console.WriteLine("Invalid data type");
         return false;
      }
   }
   public string[] CalculatePrice()
```

```
    {
        string[] str=new string[5];
        double TotalPrice=Quantity*PricePerKg;
        if(Flavour=="Vanila")
        {
            TotalPrice=TotalPrice-(TotalPrice)*0.05;
        }
        else if(Flavour=="Chocolate")
        {
            TotalPrice=TotalPrice-(TotalPrice)*0.1;
        }
        else
        {
            TotalPrice=TotalPrice-(TotalPrice)*0;
        }
        str[0]=CakeType;
        str[1]=Flavour;
        str[2]=Quantity.ToString();
        str[3]=PricePerKg.ToString();
        str[4]=TotalPrice.ToString();
        return str;

    }
}
public class Program
{
    static void Main()
    {
        CakeUtility c=new CakeUtility();
        Console.WriteLine("Enter the cake type");
        c.CakeType=Console.ReadLine();
```

```
        if(c.ValidateCakeType())
        {
            Console.WriteLine("Enter the flavour");
            c.Flavour=Console.ReadLine();
            Console.WriteLine("Enter the Quantity");
            c.Quantity=int.Parse(Console.ReadLine());
            Console.WriteLine("Enter the PricePerKg");
            c.PricePerKg=int.Parse(Console.ReadLine());
            string[] arr=c.CalculatePrice();
            Console.WriteLine("Cake type : "+ arr[0]);
            Console.WriteLine("Cake flavour : "+arr[1]);
            Console.WriteLine("Quantity : "+arr[2]);


        }

    }
}
```

---

6)plant heaven

```
using System;
public class Plant
{
    public string PlantName{get;set;}
    public int NoOfSapling{get;set;}
    public string Category{get;set;}
    public int PricePerSapling{get;set;}
```

```
    public Plant()
    {

    }
}
public class PlantUtility:Plant
{
    public Plant ExtractDetails(string plantDetails)
    {
        string[] k=plantDetails.Split(":");
        this.PlantName=k[0];
        this.NoOfSapling=int.Parse(k[1]);
        this.Category=k[2];
        this.PricePerSapling=int.Parse(k[3]);
        return this;

    }
    public double CalculateCost()
    {
       double total=NoOfSapling*PricePerSapling;
        if(total>500 && total<=1000)
        {
            total=total-(total)*0.1;
        }
        else if(total>1000)
        {
            total=total-(total)*0.2;
        }
        return total;
    }
```

```
}
public class Program
{
   static void Main()
   {
      PlantUtility pu=new PlantUtility();
      Console.WriteLine("Enter the plant details");
      string k=Console.ReadLine();
      pu.ExtractDetails(k);
      Console.WriteLine($"Plant Name is {pu.PlantName}\n No Of Sapling {pu.NoOfSapling}\n
Category {pu.Category}\n PricePerSapling is {pu.PricePerSapling}");
      Console.WriteLine("total Cost is :"+pu.CalculateCost());

   }
}
```

---

7)carslon bike insurance

```
using System;
using System.Collections.Generic;
public class Bike
{
   public string BikeNumber{get;set;}
   public int EngineCapacity{get;set;}
   public double Year{get;set;}
   public double Cost{get;set;}
   public double Insurance{get;set;}
```

```
}
public class BikeUtility:Bike
{
   public bool ValidateBikeNumber()
   {

if((BikeNumber.Length==5)&&(char.IsUpper(this.BikeNumber[0]))&&(char.IsUpper(this.BikeNumber
[1]))&&(char.IsDigit(this.BikeNumber[2]))&&(char.IsDigit(this.BikeNumber[3]))&&(char.IsDigit(this.Bi
keNumber[4])))
      {
         return true;
      }
      else
      {
         return false;
      }
   }
   public Bike CalculateInsurance()
   {

      if(EngineCapacity<=200)
      {
         if(Year<=2000)
         {
            Insurance=Cost*0.01;
         }
         else
         {
            Insurance=Cost*0.02;
         }
      }
```

```
        else
        {
            if(Year<=2000)
            {
                Insurance=Cost*0.03;
            }
            else
            {
                Insurance=Cost*0.04;
            }
        }
        return this;

    }
}
public class Program
{
    static void Main(string[] args)
    {
        BikeUtility b=new BikeUtility();
        Console.WriteLine("Enter the bike number");
        b.BikeNumber=Console.ReadLine();
        if(b.ValidateBikeNumber())
        {
            Console.WriteLine("Enter the Engine Capacity");
            b.EngineCapacity=int.Parse(Console.ReadLine());
            Console.WriteLine("Enter the Year");
            b.Year=double.Parse(Console.ReadLine());
            Console.WriteLine("Enter the cost of the bike");
            b.Cost=double.Parse(Console.ReadLine());
            b.CalculateInsurance();
```

```
        Console.WriteLine("BikeNumber\tEngineCapacity\tYear\tCost\tInsurance");
        Console.WriteLine($"{b.BikeNumber}\t\t {b.EngineCapacity}\t\t {b.Year}\t {b.Cost}\t {b.Insurance}");

      }
      else
      {
        Console.WriteLine("Enter invalid");
      }


   }
}
```

---

8)madoff's minions

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Collections;
public class Program
{
   public static Dictionary<string,float> empDictionary=new Dictionary<string,float>();
   public static void Main(string[] args)
   {
      empDictionary.Add("EMP101",2.5f);
      empDictionary.Add("EMP102",4.3f);
      empDictionary.Add("EMP103",5.0f);
      empDictionary.Add("EMP104",3.4f);
      empDictionary.Add("EMP105",6.0f);
```

```
Program pr=new Program();
while(true)
{
   Console.WriteLine("Enter your choice");
   Console.WriteLine("1. Find Employee Rating\n2. Find Employee With Highest Rating\n3. Sort Employees By Rating\n4. Exit");
   int choice=int.Parse(Console.ReadLine());
   switch(choice)
   {
      case 1:
         Console.WriteLine("Enter the employee id");
         string k=Console.ReadLine();
         float res=pr.FindEmployeeRating(k);
         if(res==-1)
         {
            Console.WriteLine("Invalid employee id");
         }
         else
         {
            Console.WriteLine("Rating is : "+res);
         }
         break;
         case 2:
            Console.WriteLine("Enter the Rating");
            float rating=float.Parse(Console.ReadLine());
            List<string> list=pr.FindEmployeeWithHighestRating(rating);
            if(list.Count>0)
            {
               foreach(string item in list)
               {
                  Console.WriteLine(item);
```

```
                }
            }
            else
            {
                Console.WriteLine("No highest rating employees found");
            }
            break;
        case 3:
            Dictionary<string,float> mydict=pr.SortByRating();
            Console.WriteLine("EmployeeId\tRating");
            foreach(var item in mydict)
            {
                Console.WriteLine($"{item.Key}\t\t{item.Value}");
            }
            break;
        case 4:
            Console.WriteLine("Thank you");
            return;
    }

  }

}
public float FindEmployeeRating(string empid)
{
  float res1=-1;
  foreach(var item in empDictionary)
  {
    if(item.Key==empid)
    {
      res1=item.Value;
```

```
        }
      }
      return res1;

   }
   public List<string> FindEmployeeWithHighestRating(float rating)
   {
      List<string> k=new List<string>();
      foreach(var item in empDictionary)
      {
         if(rating==item.Value)
         {
            k.Add(item.Key);
         }
      }
      return k;
   }
   public Dictionary<string,float> SortByRating()
   {
      return empDictionary.OrderByDescending(c=>c.Value).ToDictionary(a=>a.Key,a=>a.Value);
   }

}
```

---

9)win cineams

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
public class Movie
{
   public string MovieName{get;set;}
   public string ScreenedDate{get;set;}
   public string RemovedDate{get;set;}
   public double Price{get;set;}
}

public class Program
{
   public static Dictionary<int,Movie> screeningDetails = new Dictionary<int,Movie>();

   public Dictionary<string,double>MovieScreenedMoreNumberOfDays()
   {

      Dictionary<string,double> result=new Dictionary<string,double>();
      double maxScreenDays=0;
      foreach(var movieEntry in screeningDetails)
      {
         DateTime screenedDate=DateTime.Parse(movieEntry.Value.ScreenedDate);
         DateTime removedDate=DateTime.Parse(movieEntry.Value.RemovedDate);
         double screenedDays=(removedDate-screenedDate).TotalDays;
         if(screenedDays>maxScreenDays)
         {
```

```
            maxScreenDays=screenedDays;
            result.Clear();
            result.Add(movieEntry.Value.MovieName,movieEntry.Value.Price);
        }
        else if(screenedDays==maxScreenDays)
        {
            result.Add(movieEntry.Value.MovieName,movieEntry.Value.Price);
        }
    }
    return result;
}
public Dictionary<string,double> MovieWithScreenedDays()
{
    Dictionary<string,double> result = new Dictionary<string,double>();
    foreach(var movieEntry in screeningDetails)
    {
        DateTime screenedDate=DateTime.Parse(movieEntry.Value.ScreenedDate);
        DateTime removedDate=DateTime.Parse(movieEntry.Value.RemovedDate);
        double screenedDays = (removedDate-screenedDate).TotalDays;
        result.Add(movieEntry.Value.MovieName,screenedDays);
    }
    return result;
}
static void Main()
{
    Program pr=new Program();
    screeningDetails.Add(1,new Movie {MovieName="Eternals",ScreenedDate="04/25/2020",
RemovedDate="05/30/2020",Price=350});
    screeningDetails.Add(2,new Movie {MovieName="Iron Man",ScreenedDate="07/15/2008",
RemovedDate="08/15/2008",Price=1350});
    screeningDetails.Add(3,new Movie {MovieName="Avatar",ScreenedDate="10/15/2003",
RemovedDate="02/05/2004",Price=3500});
```

```
    screeningDetails.Add(4,new Movie {MovieName="Light Year",ScreenedDate="05/17/2020",
RemovedDate="07/03/2020",Price=6350});
    while(true)
    {
      Console.WriteLine("1. Movie Screening More number of days");
      Console.WriteLine("2. Movie with their Screening Days");
      Console.WriteLine("3. Exit");
      Console.WriteLine("Enter your choice");
      int choice=int.Parse(Console.ReadLine());
      switch(choice)
      {
        case 1:
          Dictionary<string,double> mydict1=pr.MovieScreenedMoreNumberOfDays();
          foreach(var item1 in mydict1)
          {
            Console.WriteLine($"{item1.Key} {item1.Value}");
          }
          break;
        case 2:
          Dictionary<string,double> mydict2=pr.MovieWithScreenedDays();
          foreach(var item2 in mydict2)
          {
            Console.WriteLine($"{item2.Key} {item2.Value}");

          }
          break;
        case 3:
          Console.WriteLine("Thank you");
          return;

      }
```

```
      }


   }
}
```

---

10)revv bikes

```
using System;
using System.Collections.Generic;
public class Program
{
   public static void Main()
   {
      Program pr=new Program();
      while(true)
      {
         Console.WriteLine("1. Add Bike Details\n2. View Count By cubic Capacity\n3. View Bikes By cubic Capacity\n4. Exit");
         Console.WriteLine("Enter the choice");
         int choice=int.Parse(Console.ReadLine());
         switch(choice)
         {
            case 1:
               Console.WriteLine("Enter the number of entries");
               int en=int.Parse(Console.ReadLine());
               string[] ent=new string[en];
               for(int i=0;i<en;i++)
```

```
    {
        ent[i]=Console.ReadLine();
    }
    pr.AddBikeDetails(ent);
    break;
case 2:
    Console.WriteLine("Enter the cubic Capacity");
    int cc=int.Parse(Console.ReadLine());
    if(pr.ViewBookedCount(cc)!=-1)
    {
        Console.WriteLine("Booked Count: "+pr.ViewBookedCount(cc));
    }
    else
    {
        Console.WriteLine("No bikes are booked");
    }
    break;
case 3:
    Console.WriteLine("Enter the cubic Capacity");
    int cc1=int.Parse(Console.ReadLine());
    List<string> bikename=pr.ViewBikeNames(cc1);
    if(bikename.Count>0)
    {
        foreach(var item in bikename)
        {
            Console.WriteLine(item);
        }
    }
    else
    {
        Console.WriteLine("no bikes are booked");
```

```
            }
            break;
        case 4:
            Console.WriteLine("Thank you");
            return;


    }
  }

}
public static Dictionary<string,int> BikeDetails = new Dictionary<string,int>();

public void AddBikeDetails(string[] bike)
{
  foreach(var item in bike)
  {
    string[] details=item.Split("_");
    BikeDetails.Add(details[0],Convert.ToInt32(details[1]));

  }
}
public int ViewBookedCount(int cubicCapacity)
{
  int count=0;
  foreach(var item in BikeDetails)
  {
    if(cubicCapacity==item.Value)
    {
      count++;
```

```
            }
        }
        if(count>0)
        {
            return count;
        }
        else
        {
            return -1;
        }
    }
    public List<string> ViewBikeNames(int cubicCapacity)
    {
        List<string> bikename=new List<string>();
        foreach(var item in BikeDetails)
        {
            if(cubicCapacity==item.Value)
            {
                bikename.Add(item.Key);
            }
        }
        return bikename;
    }
}
```

---

11)secret trade

```
using System;
using System.Collections.Generic;
using System.Linq;
public class Course
{
   public string Code{get;set;}
   public string Name{get;set;}
   public int Duration{get;set;}
   public double Fee{get;set;}



}
public class Program
{
   public static void Main()
   {
      Courses.Add(new Course{Code="C01",Name="C", Duration=25,Fee=12000});
      Courses.Add(new Course{Code="c02",Name="C++",Duration=30,Fee=15000});
      Courses.Add(new Course{Code="c03",Name="Java",Duration=60,Fee=45000});
      Courses.Add(new Course{Code="c04",Name="C Sharp",Duration=50,Fee=30000});
      Courses.Add(new Course{Code="c05",Name="Python",Duration=40,Fee=18000});

      Program pr=new Program();
      while(true)
      {
         Console.WriteLine("1. Find the course duration");
         Console.WriteLine("2. Update Course fee");
         Console.WriteLine("3. Sort course by fee");
         Console.WriteLine("4. Exit");
         Console.WriteLine("Enter your choice");
```

```
int choice=int.Parse(Console.ReadLine());
switch(choice)
{
   case 1:
      Console.WriteLine("Enter the course name");
      string k=Console.ReadLine();
      Dictionary<string,int> arr=pr.FindTheCourseDuration(k);
      foreach(var item in arr)
      {
         Console.WriteLine(item.Key+" "+item.Value);
      }
      break;
   case 2:
         Console.WriteLine("Enter the code");
      string cd=Console.ReadLine();
      Console.WriteLine("Enter the amount");
      double am=Convert.ToDouble(Console.ReadLine());
      Course cr=pr.UpdateCourseFee(cd,am);
      if(cr==null)
      {
         Console.WriteLine("not found");
      }
      else
      {
         Console.WriteLine("Code: "+cr.Code);
         Console.WriteLine("Name: "+cr.Name);
         Console.WriteLine("Duration: "+cr.Duration);
      }
      break;
   case 3:
      List<string> list = pr.SortCourseByFee();
```

```
            foreach(var item in list)
            {
                Console.WriteLine(item);
            }
            break;
        case 4:
            Console.WriteLine("Thank you");
            return;
        default:
            Console.WriteLine("Invalid ");
            break;


    }
  }

}
public static List<Course> Courses=new List<Course>();

public Dictionary<string,int> FindTheCourseDuration(string courseName)
{
  Dictionary<string,int> mydict=new Dictionary<string,int>();
  foreach(var item in Courses)
  {
    if(item.Name==courseName)
    {
      mydict.Add(item.Name,item.Duration);
    }
  }
  return mydict;
}
```

```
    public Course UpdateCourseFee(string code,double amount)
    {
        foreach(var item in Courses)
        {
            if(item.Code==code)
            {
                item.Fee=amount;
                return item;
            }
        }
        Course course=null;
        return course;
    }
    public List<string> SortCourseByFee()
    {
        return Courses.OrderByDescending(c=>c.Fee).Select(c=>c.Name).ToList();
    }
}
```

---

12)elegant jewllers

```
using System;
public class Bill
    {
        public string MetalName {  get; set; }
        public double Weight {  get; set; }
        public double PurityOfMetal {  get; set; }
```

```
        public bool WantDecoration { get; set; }
    }
public class Service:Bill
{
    public void ExtractDetails(string billDetails)
    {
        string[] details=billDetails.Split(":");
        MetalName=details[0];
        Weight=double.Parse(details[1]);
        PurityOfMetal=double.Parse(details[2]);
        WantDecoration=bool.Parse(details[3]);
    }
    public bool ValidateMetalName()
    {
        if(MetalName=="Gold"||MetalName=="Silver")
        {
            return true;
        }
        else
        {
            return false;
        }
    }
    public double CalculateTotalPrice()
    {
        double total=0;
        if(MetalName=="Gold")
        {
            total=5000*(PurityOfMetal/100)*Weight;
        }
        else if(MetalName=="Silver")
```

```
        {
            total=100*(PurityOfMetal/100)*Weight;
        }
        if(WantDecoration)
        {
            total=total+((total/100)*10);
        }
        return total;
    }
}
public class program
{
    static void Main(string[] args)
    {
        Service obj=new Service();
        Console.WriteLine("Enter the bill details");
        string user=Console.ReadLine();
        obj.ExtractDetails(user);
        if(obj.ValidateMetalName())
        {
            Console.WriteLine("The bill amount is "+obj.CalculateTotalPrice());
        }
        else
        {
            Console.WriteLine("Invalid Metal Name");
        }
    }
}
```

---

13)destuche bank

```
using System;
public class Customer
{
   public string CustomerName{get;set;}
   public long SSn{get;set;}
   public string City{get;set;}
   public double LoanAmount{get;set;}
   public int NoOfYears{get;set;}

   public Customer()
   {

   }

}
public class CustomerUtility:Customer
{
   public CustomerUtility(string name, long ssn, string city, double loan, int years)
   {
         CustomerName = name;
         SSn = ssn;
         City = city;
         LoanAmount = loan;
         NoOfYears = years;
   }
   public string GenerateTokenNumber()
```

```
    {


        //return CustomerName.Substring(0,2).ToUpper()+" "+City.Substring(0,2).ToUpper()+"
"+SSn%100;
        return String.Concat(CustomerName.Substring(0, 2).ToUpper(), City.Substring(2, 1).ToUpper(),
SSn % 100);
    }
    public double CalculateAnnualInterest(string loanType)
    {
        double au=0;
        if(loanType=="Home")
        {
            au=LoanAmount*0.03*NoOfYears;
        }
        else if(loanType=="Business")
        {
            au=LoanAmount*0.05*NoOfYears;
        }
        else if(loanType=="Gold")
        {
            au=LoanAmount*0.02*NoOfYears;
        }
        return au;
    }
}
public class Program
{
    static void Main(string[] args)
    {


        Console.WriteLine("Enter the cutsomerName");
        string name=Console.ReadLine();
```

```
        Console.WriteLine("Enter SSn");
        long ssn=long.Parse(Console.ReadLine());
        Console.WriteLine("Enter city");
        string city=Console.ReadLine();
        Console.WriteLine("Enter the total loan amount");
        double loan=double.Parse(Console.ReadLine());
        Console.WriteLine("Enter the number of years");
        int years=int.Parse(Console.ReadLine());
        Console.WriteLine("Enter the loan type");
        string type=Console.ReadLine();
        CustomerUtility obj = new CustomerUtility(name, ssn, city,loan, years);
        Console.WriteLine(obj.GenerateTokenNumber());
        Console.WriteLine(obj.CalculateAnnualInterest(type));
    }
}
```

