Create a hierarchy of Employee, Manager, MarketingExecutive in Employee Management System. They should have the following functionality.

Manager with following private members.

Petrol Allowance: 8 % of Salary.

Food Allowance : 13 % of Salary.

Other Allowances : 3% of Salary.

Calculate GrossSalary by adding above allowances. Override CalculateSalary() method to calculate Net Salary. NetSalary. PF calculation should not consider above allowances.

MarketingExecutive with following private members.Kilometer travel

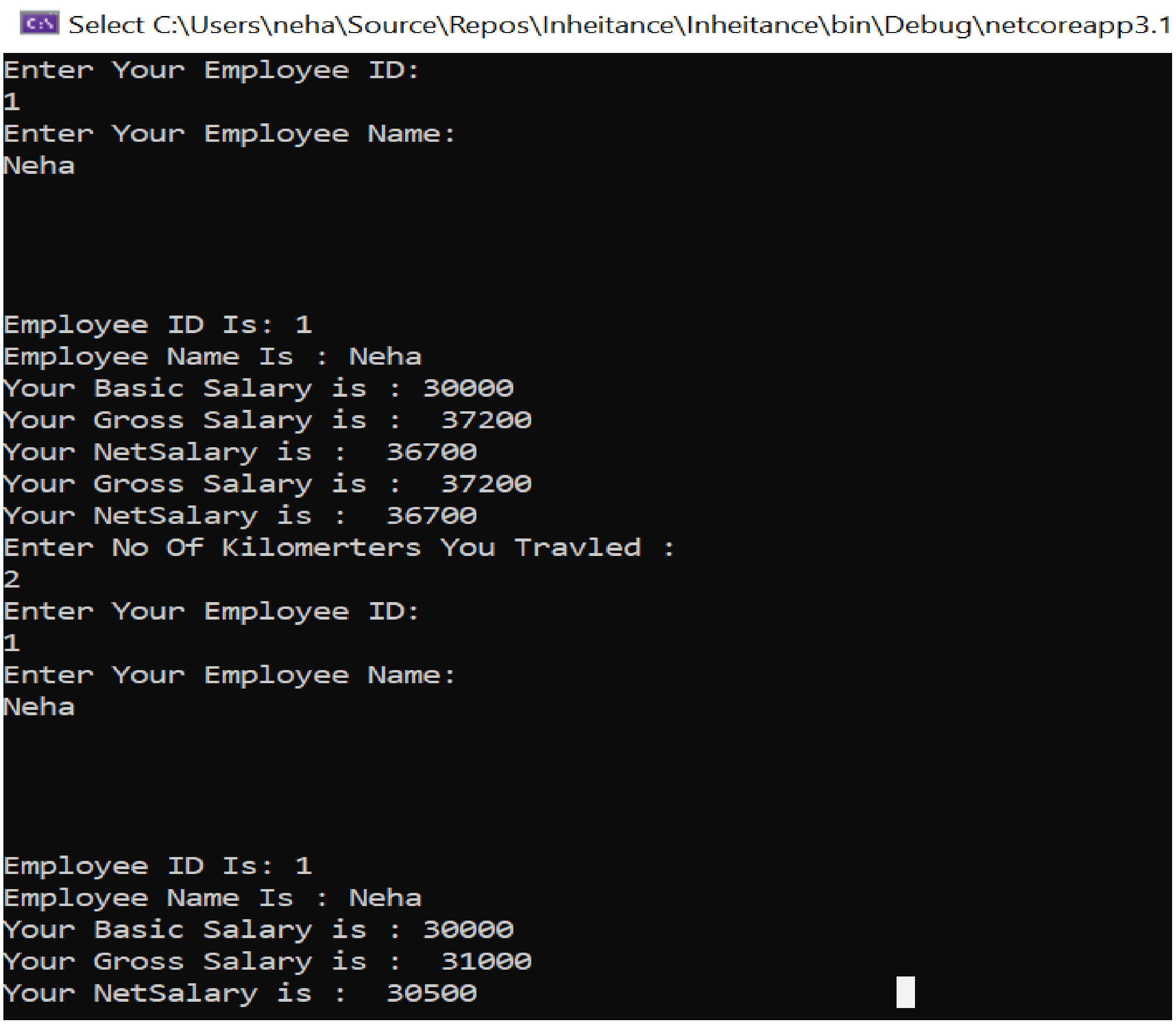
Tour Allowances : Rs 5/- per Kilometer (Automatically generated).

Telephone Allowances : Rs.1000/-

Calculate GrossSalary by adding above allowances. Override CalculateSalary(). NetSalary,PF calculation should not consider above allowances.

Ans:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace csharpAssignments3
{




    internal class DemoEmp
  {
    public int empid;
    public string empname;
    public double basic_salary = 30000;
    public double km;


      public void getinfo()
      {
        Console.WriteLine("Enter Your Employee ID: ");
        empid = Convert.ToInt32(Console.ReadLine());

        Console.WriteLine("Enter Your Employee Name: ");
        empname = Console.ReadLine();


      }
      public void getkmdetails()
      {
        Console.WriteLine("Enter No Of Kilomerters You Travled :");
        km = Convert.ToDouble(Console.ReadLine());

      }
      public void show()
      {
            Console.WriteLine("\n\n\n");
            Console.WriteLine("Employee ID Is: " + empid);
            Console.WriteLine("Employee Name Is : " + empname);

        }

        public void CalculateSalary()
        {
            Console.WriteLine("Your Basic Salary is : " + basic_salary);
        }

    }

    class Manager : DemoEmp
    {

        public void getempdetails()
        {
            getinfo();
            base.show();
            base.CalculateSalary();


        }
        public void CalculateSalary()
        {
            double netsal;
            double grossal;
            grossal = base.basic_salary + (base.basic_salary * 0.08) + (base.basic_salary * 0.13) +
(base.basic_salary * 0.03);
            Console.WriteLine("Your Gross Salary is :  " + grossal);
            netsal = grossal - 500;//500 reduse as PF
            Console.WriteLine("Your NetSalary is :  " + netsal);

        }
    }
    class Marktingexc : DemoEmp
    {
        public void getempdetails()
        {
            base.getinfo();

            base.show();
            base.CalculateSalary();



            ;
        }
        public void CalculateSalary()
        {
            double netsal;
            double grossal;
            grossal = base.basic_salary + (base.km / 5) + 1000;
            Console.WriteLine("Your Gross Salary is :  " + grossal);
            netsal = grossal - 500;//500 reduse as PF
            Console.WriteLine("Your NetSalary is :  " + netsal);

        }
    }


    class Program
    {
        static void Main(string[] args)
        {

            Manager mg = new Manager();
            mg.getempdetails();
            mg.CalculateSalary();
            mg.CalculateSalary();
            mg.getkmdetails();



            Marktingexc mk = new Marktingexc();

            mk.getempdetails();
            mk.CalculateSalary();


            Console.ReadKey();
        }
    }

}
```
Output:


Select C:\Users\neha\Source\Repos\Inheitance\Inheitance\bin\Debug\netcoreapp3.1

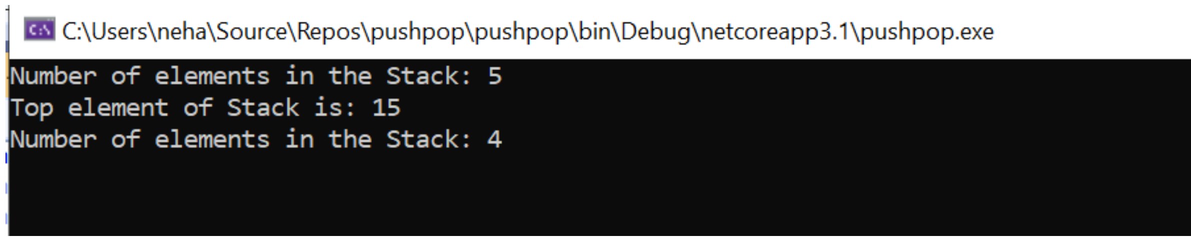
```
Enter Your Employee ID:
1
Enter Your Employee Name:
Neha




Employee ID Is: 1
Employee Name Is : Neha
Your Basic Salary is : 30000
Your Gross Salary is :  37200
Your NetSalary is :  36700
Your Gross Salary is :  37200
Your NetSalary is :  36700
Enter No Of Kilomerters You Travled :
2
Enter Your Employee ID:
1
Enter Your Employee Name:
Neha




Employee ID Is: 1
Employee Name Is : Neha
Your Basic Salary is : 30000
Your Gross Salary is :  31000
Your NetSalary is :  30500
```

1) Write a class called `MyStack` with following members.
   a) integer array
   b) integer variable to store top position
   c) size of the array.
   Implement `Push()` and `Pop()` operation. Implement `ICloneable` interface to perform cloning. Write a client application to perform cloning.

Ans:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Collections;

namespace CustomException
{
    class myStack
    {
        public static void Main()
        {

            // Creating a Stack
            Stack myStack = new Stack();

            // Inserting the elements into the Stack
            myStack.Push("11");
            myStack.Push("12");
            myStack.Push("13");
            myStack.Push("14");
            myStack.Push("15");

            Console.WriteLine("Number of elements in the Stack: {0}",
                                          myStack.Count);

            // Retrieveing top element of Stack
            Console.Write("Top element of Stack is: ");
            Console.Write(myStack.Pop());

            // printing the no of Stack element
            // after Pop operation
            Console.WriteLine("\nNumber of elements in the Stack: {0}",
                                            myStack.Count);
            Console.ReadKey();
        }
    }
                    }
```

Output:



```
C:\Users\neha\Source\Repos\pushpop\pushpop\bin\Debug\netcoreapp3.1\pushpop.exe
Number of elements in the Stack: 5
Top element of Stack is: 15
Number of elements in the Stack: 4
```

Create a custom exception class named StackException. The Push()and Pop() method should throw object of StackException when the stack is full or empty respectively.

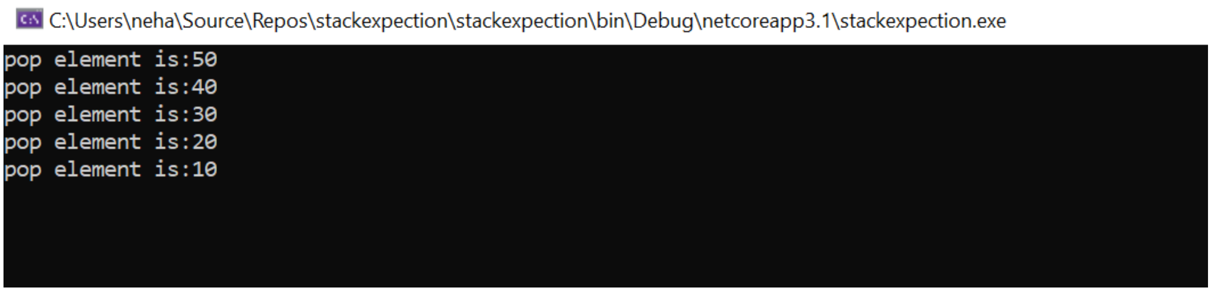
Ans:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace exception
{
    internal class Stack
    {
```

```
    private int[] ele;
    private int top;
    private int max;
    public Stack(int size)
    {
        ele = new int[size];
        top = -1;
        max = size;
    }
    public void Push(int item)
    {
        if(top == max -1)
        {
            throw new Exception("Stavk overflow not perform push");

        }
        else
        {
            ele[++top] = item;

        }
    }
    public int Pop()
    {
        if(top==-1)
        {
            throw new Exception("stack is empty");

        }
        else
        {
            Console.WriteLine("pop element is:" + ele[top]);
            return ele[top--];
        }
    }
    public void printStack()
    {
        if(top== -1)
        {
            Console.WriteLine("stack is empty");
            return;
        }
        else
        {
            for(int i=0;i<=top;i++)
            {
                Console.WriteLine("Item[" + (i + 1) + "]:"+ele);

            }
        }
    }
}
class Program
{
    public static void Main()
```

```
        {
            Stack S = new Stack(5);
            S.Push(10);
            S.Push(20);
            S.Push(30);
            S.Push(40);
            S.Push(50);
            //S.Push(60);

            Console.WriteLine("item are:");
            S.printStack();
            S.Pop();
            S.Pop();
            S.Pop();
            S.Pop();
            S.Pop();
            Console.ReadKey();

        }
    }
}
```

Output:



C:\Users\neha\Source\Repos\stackexpection\stackexpection\bin\Debug\netcoreapp3.1\stackexpection.exe

```
pop element is:50
pop element is:40
pop element is:30
pop element is:20
pop element is:10
```