Name : Anjali Manishbhai Hingu

Prn : 2020033800098732

**Assignment – 6**

Question 1:

**Code:**

class Program

{

static public void Sort<T>(IList<T> sortArray, Func<T, T, bool> compare)

{

for (int i = 0; i < sortArray.Count - 1; i++)

{

for (int j = i + 1; j < sortArray.Count; j++)

{

if (compare(sortArray[j], sortArray[i]))

{

T temp = sortArray[i];

sortArray[i] = sortArray[j]; sortArray[j] = temp;

}

}

}

}

}

public enum Position

{

Peon = 1,

Assistant\_Proffesor = 2,

Permanent\_Professor = 3,

HOD = 4,

Dean = 5,

Vice\_chancellor = 6

}

class University

{

private int id; private float salary; private string name; private Position position;

public int ID

{

get { return id; }

set { id = value; }

}

public float Salary

{

get { return salary; }

set { salary = value; }

}

public string Name

{

get { return name; }

set { name = value; }

}

public Position Position

{

get { return position; }

set { position = value; }

}

public University(int Id, float Salary, string Name, Position position)

{

this.id = Id; this.salary = Salary; this.name = Name; this.position = position;

}

internal static bool CompareSalary(University u1, University u2)

{

return u1.salary < u2.salary;

}

internal static bool ComparePosition(University u1, University u2)

{

return u1.position > u2.position;

}

}

public class TestGenericMethods

{

public static void Main(string[] args)

{

List<University> list = new List<University>(10);

list.Add(new University(6, 15000, "Peon", Position.Peon)); list.Add(new University(5, 30000, "Assi\_Proff", Position.Assistant\_Proffesor)); list.Add(new University(2, 100000, "Dean", Position.Dean));

list.Add(new University(1, 225000, "VC", Position.Vice\_chancellor)); list.Add(new University(4, 50000, "Per\_Proff", Position.Permanent\_Professor)); list.Add(new University(3, 80000, "Hod", Position.HOD));

Program.Sort<University>(list, University.CompareSalary); Console.WriteLine("Sorting On Salary : "); foreach (University u in list)

{

Console.WriteLine($"{u.ID}) {u.Name} , {u.Position} {u.Salary}");

}

Program.Sort<University>(list, University.ComparePosition); Console.WriteLine("Sorting On Position : "); foreach (University u in list)

{

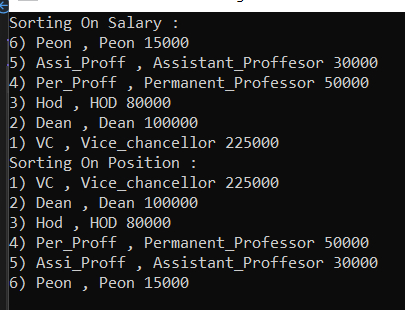
Console.WriteLine($"{u.ID}) {u.Name} , {u.Position} {u.Salary}");

}

}

}

**Output:**



Question 2:

**Code:**

using System.Security.Principal;

public delegate string Delegate();

class Account

{

static int accountNo; static int balance;

public Account(int accNo, int bal)

{

accountNo = accNo; balance = bal;

}

public int AccountNo

{

get => accountNo;

}

public int Balance

{

get => balance;

}

public static string Msg()

{

if (balance < 0)

{

return "you are overdrawn";

}

else if (balance < 10)

{

return "your account is very low";

}

else if (balance < 100)

{

return "watch your spending carefully";

}

else

{

return "you have over $100 in your account";

}

}

}

class Programme

{

static void Main(string[] args)

{

Delegate msg = Account.Msg;

Account a1 = new Account(120000000, 10000);

Console.WriteLine($"{a1.AccountNo} => {a1.Balance} Message:{msg()}"); Account a2 = new Account(122134562, 7);

Console.WriteLine($"{a2.AccountNo} => {a2.Balance} Message:{msg()}"); Account a3 = new Account(267891000, 60);

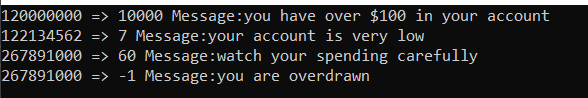
Console.WriteLine($"{a3.AccountNo} => {a3.Balance} Message:{msg()}"); Account a4 = new Account(267891000, -1);

Console.WriteLine($"{a4.AccountNo} => {a4.Balance} Message:{msg()}");

}

}

**Output:**



Question 1:

**Code:**

class Program

{

public static void CallAnonymousMethod()

{

bool positive = new Func<int, bool>(delegate (int int32) { return int32 > 0; })(-1); new Action<bool>(delegate (bool value) { Console.WriteLine(value); })(positive);

}

public static void CallLambda()

{

bool positive = new Func<int, bool>(int32 => int32 > 0)(1); new Action<bool>(delegate (bool value) { Console.WriteLine(value); })(positive);

}

static void Main(string[] args)

{

CallAnonymousMethod();

CallLambda();

var parse = (double x, double y) => (x > y ? x : y); double z = parse(10.1, 7); new Action<double>(delegate (double value) { Console.WriteLine(value); })(z);

}

}

**Output:**

