Practical 1: Working with basic C# and ASP .NET

A. AIM: Create an application that obtains four int values from the user and displays the product.

using System;

namespace ConsoleApp

{

class Program

{

static void Main(string[] args)

{

int num1, num2, num3, num4, prod;

Console.Write("Enter number 1: ");

num1 = int.Parse(Console.ReadLine());

Console.Write("Enter number 2: ");

num2 = int.Parse(Console.ReadLine());

Console.Write("Enter number 3: ");

num3 = int.Parse(Console.ReadLine());

Console.Write("Enter number 4: ");

num4 = int.Parse(Console.ReadLine());

prod = num1 \* num2 \* num3 \* num4;

Console.WriteLine(num1 + "\*" + num2 + "\*" + num3 + "\*" + num4 + "=" + prod);

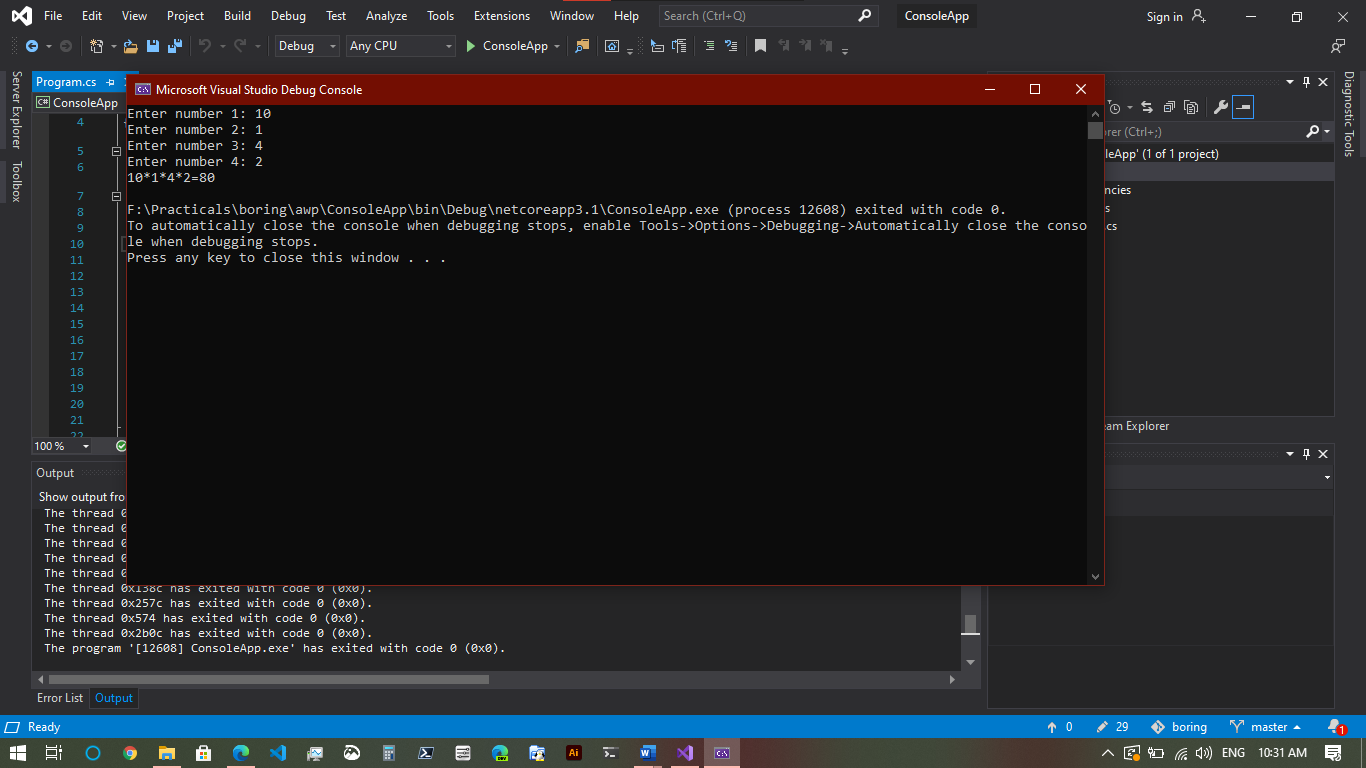
Console.ReadKey();

}

}

}

OUTPUT:



B. AIM: Create an application to demonstrate string operations.

using System;

using System.Text;

namespace StringOperate

{

class Program

{

static void Main(string[] args)

{

// Create a StringBuilder that expects to hold 50 characters.

// Initialize the StringBuilder with "ABC".

StringBuilder sb = new StringBuilder("ABC", 50);

// Append three characters (D, E, and F) to the end of the StringBuilder.

sb.Append(new char[] { 'D', 'E', 'F' });

// Append a format string to the end of the StringBuilder.

sb.AppendFormat("GHI{0}{1}", 'J', 'k');

// Display the number of characters in the StringBuilder and its string.

Console.WriteLine("{0} chars: {1}", sb.Length, sb.ToString());

// Insert a string at the beginning of the StringBuilder.

sb.Insert(0, "Alphabet: ");

// Replace all lowercase k's with uppercase K's.

sb.Replace('k', 'K');

// Display the number of characters in the StringBuilder and its string.

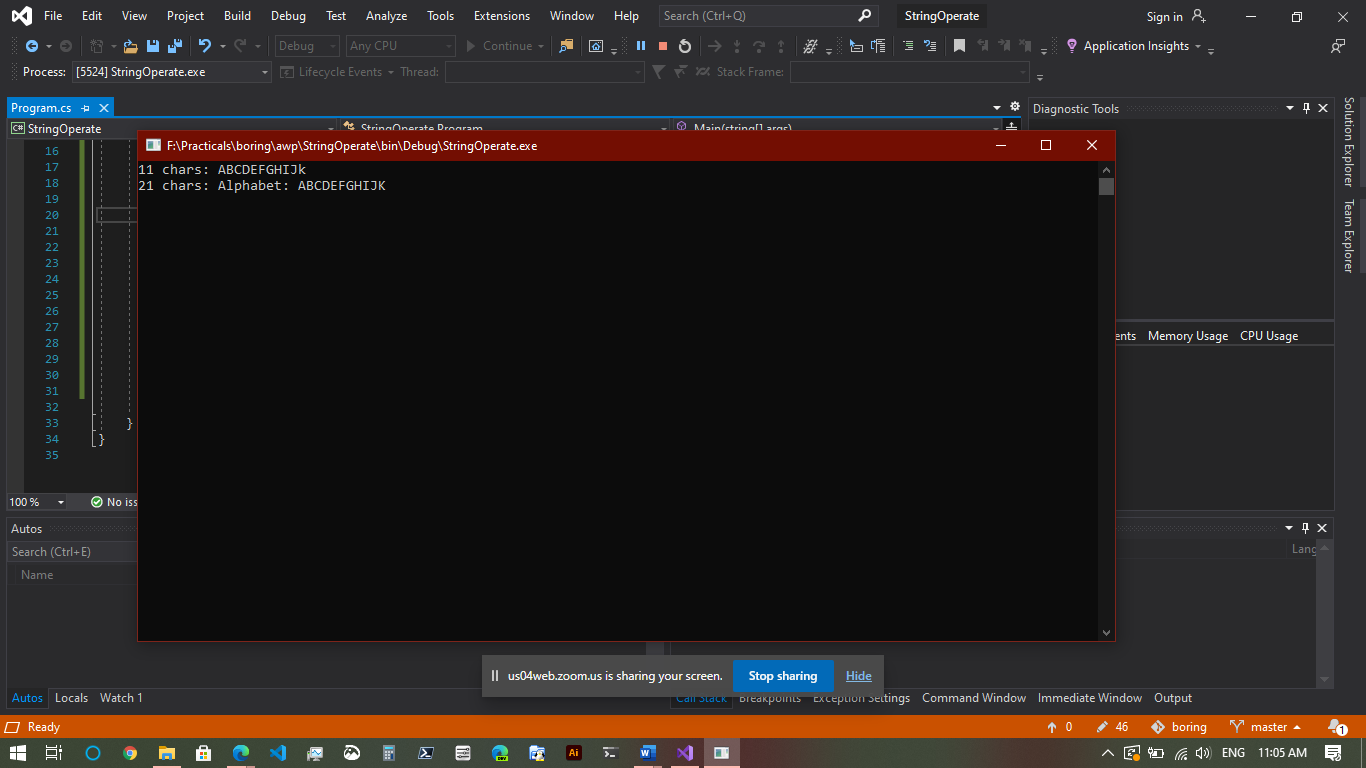
Console.WriteLine("{0} chars: {1}", sb.Length, sb.ToString());

Console.ReadKey();

}

}

}



C. AIM: Create an application that receives the (Student Id, Student Name, Course Name, Date of Birth) information from a set of students. The application should also display the information of all the students once the data entered.

using System;

namespace StudentData

{

class Program

{

struct Student

{

public string studid, name, cname;

public int day, month, year;

}

static void Main(string[] args)

{

int numberOfEntries = 2;

Student[] s = new Student[numberOfEntries];

int i;

for (i = 0; i < numberOfEntries; i++)

{

Console.Write("Enter Student Id:");

s[i].studid = Console.ReadLine();

Console.Write("Enter Student name : ");

s[i].name = Console.ReadLine();

Console.Write("Enter Course name : ");

s[i].cname = Console.ReadLine();

Console.Write("Enter date of birth\n Enter day(1-31):");

s[i].day = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter month(1-12):");

s[i].month = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter year:");

s[i].year = Convert.ToInt32(Console.ReadLine());

}

Console.WriteLine("\n\nStudent's List\n");

for (i = 0; i < numberOfEntries; i++)

{

Console.WriteLine("\nStudent ID : " + s[i].studid);

Console.WriteLine("\nStudent name : " + s[i].name);

Console.WriteLine("\nCourse name : " + s[i].cname);

Console.WriteLine("\nDate of birth(dd-mm-yy) : " + s[i].day + "-" + s[i].month +

"-" + s[i].year);

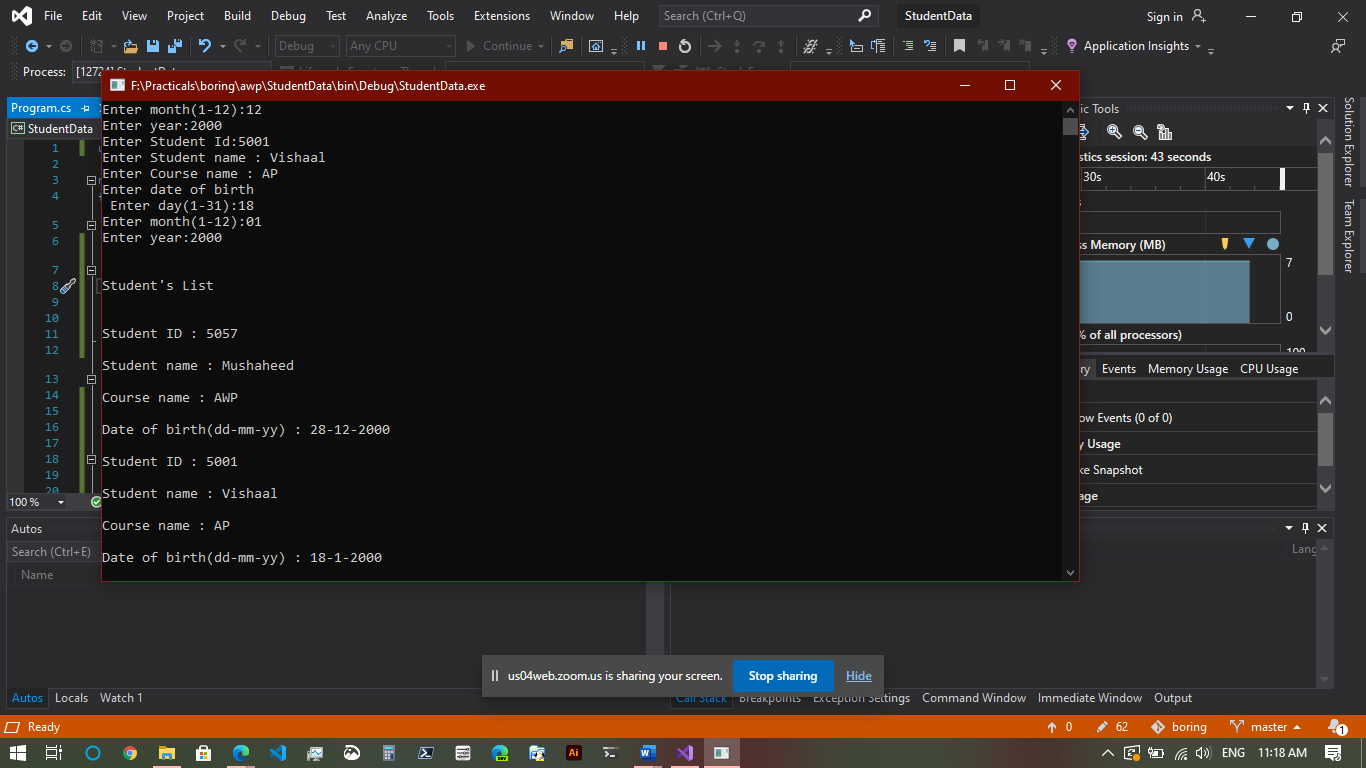
}

Console.ReadKey();

}

}

}



D. AIM: Create an application to demonstrate following operation:

i. Generate Fibonacci series, ii. Test for Prime numbers, iii. Test for vowels, iv. Use of foreach with arrays, v. Reverse a number & find sum of digits of a number.

using System;

namespace Operations1D

{

class Program

{

static void GenerateFibonacci()

{

Console.WriteLine("FIBONACCI SERIES");

int num1 = 0, num2 = 1, num3, limit, counter;

Console.Write("Upto how many number you want fibonacci series: ");

limit = int.Parse(Console.ReadLine());

counter = 3;

Console.Write(num1 + " " + num2);

while (counter <= limit)

{

num3 = num1 + num2;

if (counter >= limit)

break;

Console.Write(" " + num3);

num1 = num2;

num2 = num3;

counter++;

}

}

static void TestPrime()

{

Console.WriteLine("\n\nTest for Prime numbers");

int num, counter;

Console.Write("Enter number:");

num = int.Parse(Console.ReadLine());

for (counter = 2; counter <= num / 2; counter++)

{

if ((num % counter) == 0)

break;

}

if (num == 1)

Console.WriteLine(num + " is neither prime nor composite");

else if (counter < (num / 2))

Console.WriteLine(num + " is not prime number");

else

Console.WriteLine(num + " is prime number");

}

static void TestVowels()

{

Console.WriteLine("\nTest Vowels");

char ch;

Console.Write("Enter a character: ");

ch = (char) Console.ReadLine()[0];

switch (ch)

{

case 'a':

case 'A':

case 'e':

case 'E':

case 'i':

case 'I':

case 'o':

case 'O':

case 'u':

case 'U':

Console.WriteLine(ch + " is vowel");

break;

default:

Console.Write(ch + " is not a vowel");

break;

}

}

static void ForeachArrays()

{

Console.WriteLine("\nForeach Arrays");

string[] str = { "One", "Two", "Three" };

foreach (String s in str)

{

Console.WriteLine(s);

}

}

static void ReverseNSumNumber()

{

Console.WriteLine("\nReverse Number");

int num, actualnumber, revnum = 0, digit, sumDigits = 0;

Console.Write("Enter number: ");

num = int.Parse(Console.ReadLine());

actualnumber = num;

while (num > 0)

{

digit = num % 10;

revnum = revnum \* 10 + digit;

sumDigits = sumDigits + digit;

num = num / 10;

}

Console.WriteLine("Reverse of " + actualnumber + "=" + revnum);

Console.WriteLine("Sum of its digits: " + sumDigits);

}

static void Main(string[] args)

{

GenerateFibonacci();

TestPrime();

TestVowels();

ForeachArrays();

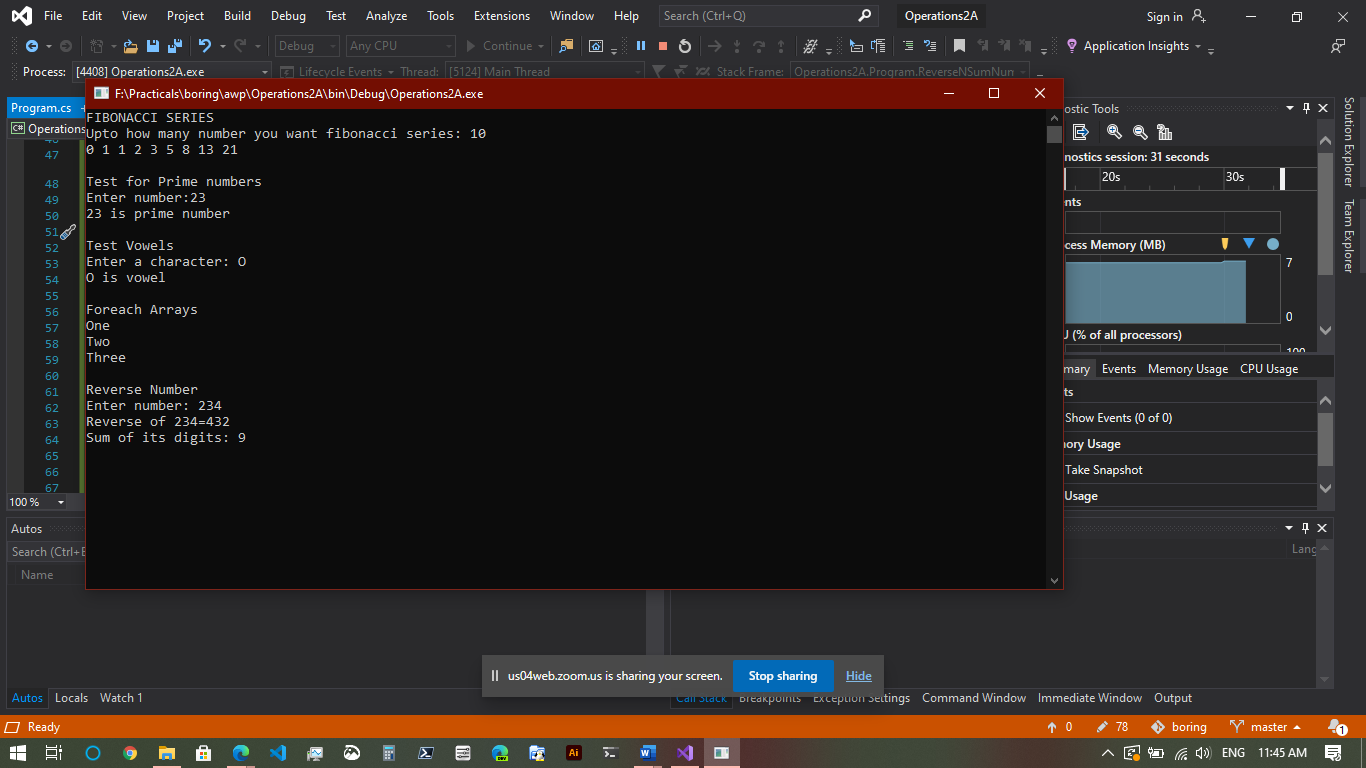
ReverseNSumNumber();

Console.ReadKey();

}

}

}



Practical 2: Working with Web Forms & Controls

A. AIM: Create simple application to perform following operations

i. Finding Factorial value, ii. Money conversion, iii. Quadratic Equation, iv. Temperature conversion

using System;

namespace Operations2A

{

class Quadraticroots

{

double a, b, c;

public void read()

{

Console.WriteLine("To find the roots of a quadratic equation of the form a\*x\*x + b\*x + c = 0");

Console.Write("Enter value for a: ");

a = double.Parse(Console.ReadLine());

Console.Write("Enter value for b: ");

b = double.Parse(Console.ReadLine());

Console.Write("Enter value for c: ");

c = double.Parse(Console.ReadLine());

}

public void compute()

{

int m;

double r1, r2, d1;

d1 = b \* b - 4 \* a \* c;

if (a == 0)

m = 1;

else if (d1 > 0)

m = 2;

else if (d1 == 0)

m = 3;

else

m = 4;

switch (m)

{

case 1:

Console.WriteLine("Not a Quadratic equation, Linear equation");

Console.ReadLine();

break;

case 2:

Console.WriteLine("Roots are Real and Distinct");

r1 = (-b + Math.Sqrt(d1)) / (2 \* a);

r2 = (-b - Math.Sqrt(d1)) / (2 \* a);

Console.WriteLine("First root is {0:#.##}", r1);

Console.WriteLine("Second root is {0:#.##}", r2);

Console.ReadLine();

break;

case 3:

Console.WriteLine("Roots are Real and Equal");

r1 = r2 = (-b) / (2 \* a);

Console.WriteLine("First root is {0:#.##}", r1);

Console.WriteLine("Second root is {0:#.##}", r2);

Console.ReadLine();

break;

case 4:

Console.WriteLine("\n Roots are Imaginary");

r1 = (-b) / (2 \* a);

r2 = Math.Sqrt(-d1) / (2 \* a);

Console.WriteLine("\n First root is {0:#.##} + i {1:#.##}", r1, r2);

Console.WriteLine("\n Second root is {0:#.##} - i {1:#.##}", r1, r2);

Console.ReadLine();

break;

}

}

}

class Program

{

static void findFactorial()

{

Console.WriteLine("Find factorial");

int i, number, fact;

Console.Write("Enter the Number: ");

number = int.Parse(Console.ReadLine());

fact = number;

for (i = number - 1; i >= 1; i--)

{

fact = fact \* i;

}

Console.WriteLine("\nFactorial of Given Number is: " + fact);

Console.ReadKey();

}

static void moneyConversion()

{

Console.WriteLine("\nMoney conversion: ");

int choice;

Console.Write("Choices:\n1 - Dollar to Rupee\n2 - Euro to Rupee\n3 - Malaysian Ringgit to Rupee\nEnter your Choice: ");

choice = int.Parse(Console.ReadLine());

switch (choice)

{

case 1:

Double dollar, rupee, val;

Console.Write("Enter the Dollar Amount: ");

dollar = Double.Parse(Console.ReadLine());

Console.Write("Enter the Dollar Value: ");

val = double.Parse(Console.ReadLine());

rupee = dollar \* val;

Console.WriteLine("{0} Dollar equals to {1} Rupees", dollar, rupee);

break;

case 2:

Double Euro, rupe, valu;

Console.Write("Enter the Euro Amount: ");

Euro = Double.Parse(Console.ReadLine());

Console.Write("Enter the Euro Value: ");

valu = double.Parse(Console.ReadLine());

rupe = Euro \* valu;

Console.WriteLine("{0} Euro equals to {1} Rupees ", Euro, rupe);

break;

case 3:

Double ringit, rup, value;

Console.Write("Enter the Ringgit Amount: ");

ringit = Double.Parse(Console.ReadLine());

Console.Write("Enter the Ringgit Value: ");

value = double.Parse(Console.ReadLine());

rup = ringit \* value;

Console.WriteLine("{0}Malaysian Ringgit equals to {1} Rupees", ringit, rup);

break;

}

}

static void quadraticRoots()

{

Console.WriteLine("\nQuadratic roots");

Quadraticroots qr = new Quadraticroots();

qr.read();

qr.compute();

}

static void temperatureConversion()

{

Console.WriteLine("\nTemperatue conversion");

int celsius, faren;

Console.Write("Enter the Temperature in Celsius(°C): ");

celsius = int.Parse(Console.ReadLine());

faren = (celsius \* 9) / 5 + 32;

Console.WriteLine("Temperature in Fahrenheit is(°F): " + faren);

}

static void Main(string[] args)

{

findFactorial();

moneyConversion();

quadraticRoots();

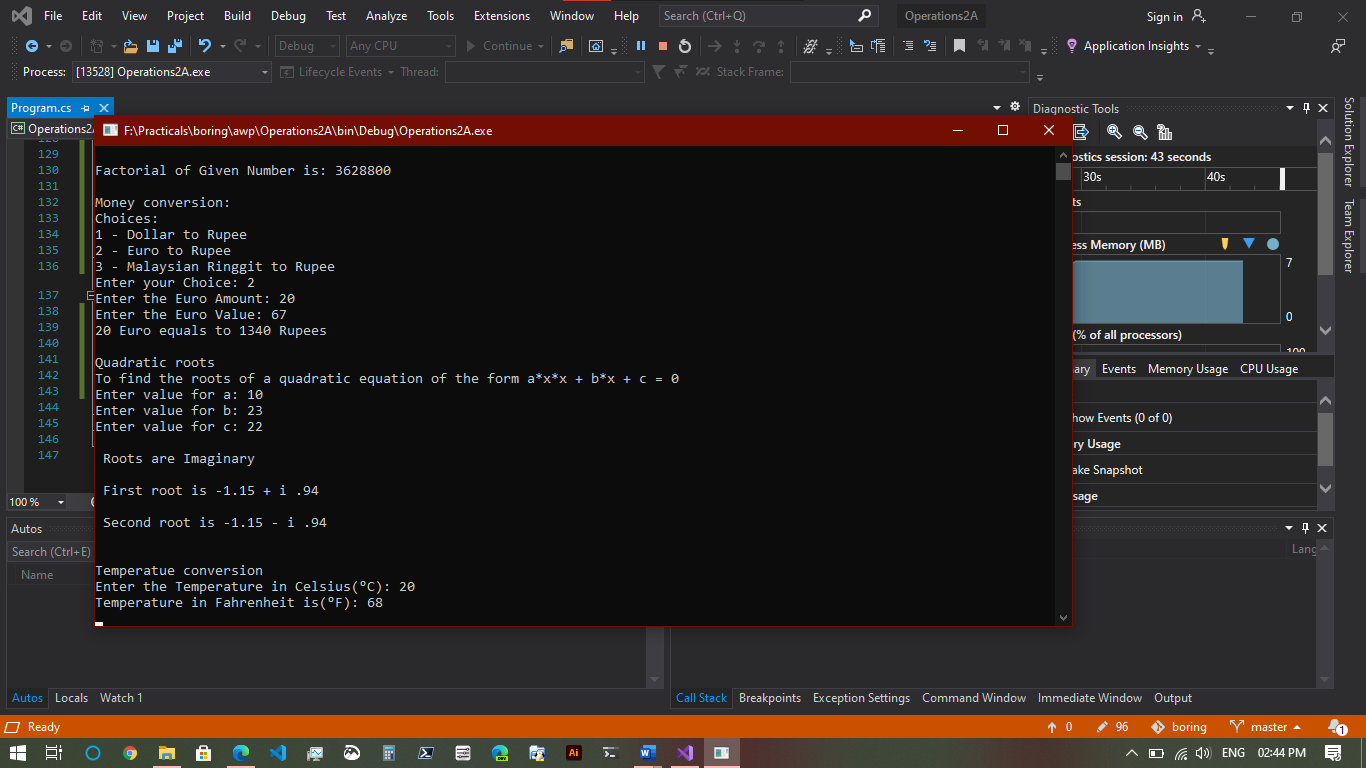
temperatureConversion();

Console.ReadKey();

}

}

}



B. AIM: Create simple application to demonstrate use of following concepts.

i. Function overloading, ii. Inheritance (All types), iii. Constructor overloading, iv. Interfaces

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Demonstrate1

{

class Overloading

{

public void swap(ref int n, ref int m)

{

int t;

t = n;

n = m;

m = t;

}

public void swap(ref float f1, ref float f2)

{

float f;

f = f1;

f1 = f2;

f2 = f;

}

}

class Furniture

{

string material;

float price;

public void getdata()

{

Console.Write("Enter material : ");

material = Console.ReadLine();

Console.Write("Enter price : ");

price = float.Parse(Console.ReadLine());

}

public void showdata()

{

Console.WriteLine("Material : " + material);

Console.WriteLine("Price : " + price);

}

}

class Table : Furniture

{

int height, surface\_area;

public void getdata()

{

base.getdata();

Console.Write("Enter height: ");

height = int.Parse(Console.ReadLine());

Console.Write("Enter surface area: ");

surface\_area = int.Parse(Console.ReadLine());

}

public void showdata()

{

base.showdata();

Console.WriteLine("Height : " + height);

Console.WriteLine("Surface Area : " + surface\_area);

}

}

class A

{

public void About()

{

Console.WriteLine("About method from class A");

}

}

class B : A

{

}

class C : B

{

}

interface D

{

void say();

}

class E : A, D

{

public void say()

{

Console.WriteLine("Inherited 'say' from an interface");

}

}

class Salary

{

int basic, ta, da, hra;

public Salary()

{

da = 9000;

hra = 6000;

}

public void getdata()

{

Console.Write("Enter basic salary : ");

basic = int.Parse(Console.ReadLine());

Console.Write("Enter travelling allowance : ");

ta = int.Parse(Console.ReadLine());

}

public void showdata()

{

Console.WriteLine("Basic salary : " + basic);

Console.WriteLine("Dearness allowence : " + da);

Console.WriteLine("Housing rent allowence : " + hra);

Console.WriteLine("Travelling allowence : " + ta);

Console.WriteLine("Gross Salary : " + (basic + da + hra + ta));

}

}

class Program

{

static void overloading()

{

Console.WriteLine("\nOverloading");

Overloading objOverloading = new Overloading();

int n = 10, m = 20;

objOverloading.swap(ref n, ref m);

Console.WriteLine("N=" + n + "\tM=" + m);

float f1 = 10.5f, f2 = 20.6f;

objOverloading.swap(ref f1, ref f2);

Console.WriteLine("F1=" + f1 + "\tF2=" + f2);

}

static void singleInheritance()

{

B obj = new B();

obj.About();

}

static void multipleInheritance()

{

E obj = new E();

obj.About();

obj.say();

}

static void multilevelInheritance()

{

C obj = new C();

obj.About();

}

static void hierarchicalInheritance()

{

B obj1 = new B();

E obj2 = new E();

obj1.About();

obj2.About();

}

static void inheritance()

{

Console.WriteLine("\nInheritance");

singleInheritance();

multipleInheritance();

multilevelInheritance();

hierarchicalInheritance();

}

static void constructorOverloading()

{

Console.WriteLine("\nConstructor overloading");

Salary s = new Salary();

s.getdata();

s.showdata();

}

static void interfaceDemo()

{

// uses interface

hierarchicalInheritance();

}

static void Main(string[] args)

{

overloading();

inheritance();

constructorOverloading();

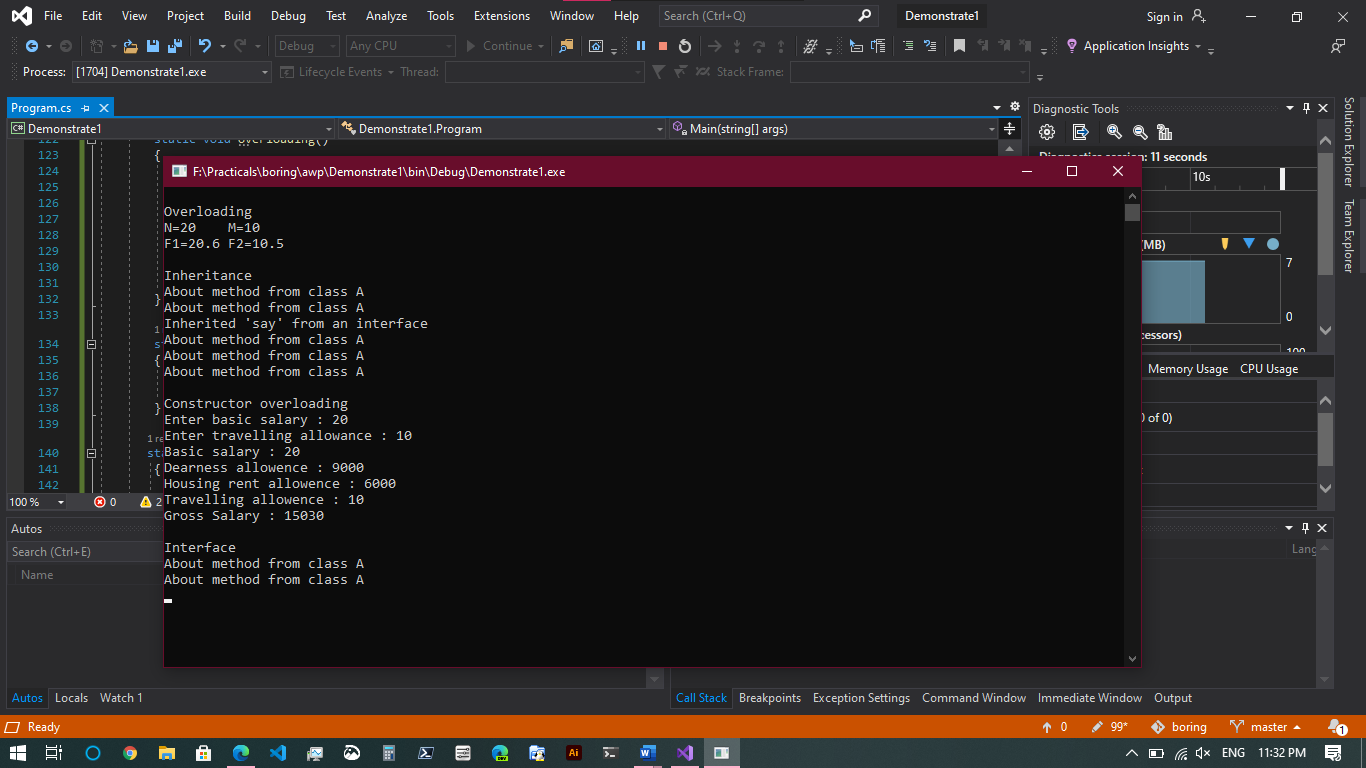
interfaceDemo();

Console.ReadKey();

}

}

}



C. AIM: Create simple application to demonstrate use of following concepts.

i. Delegates & Events, ii. Exception handling

using System;

namespace DelegateAndEvent

{

public delegate void TrafficDel();

class TrafficSignal

{

public static void Yellow()

{

Console.WriteLine("Yellow light signals to get ready");

}

public static void Green()

{

Console.WriteLine("Green light signals to go");

}

public static void Red()

{

Console.WriteLine("Red light signals to stop");

}

TrafficDel[] td = new TrafficDel[3];

public void IdentifySignal()

{

td[0] = new TrafficDel(Yellow);

td[1] = new TrafficDel(Green);

td[2] = new TrafficDel(Red);

}

public void display()

{

td[0]();

td[1]();

td[2]();

}

}

class NotEvenException : Exception

{

public NotEvenException(string msg)

: base(msg)

{

}

}

class Program

{

static void delegateExample()

{

Console.WriteLine("\nDelegate & Events example");

TrafficSignal ts = new TrafficSignal();

ts.IdentifySignal();

ts.display();

}

static void handlingExceptions()

{

Console.WriteLine("\nException handling");

int num;

try

{

Console.Write("Enter a number: ");

num = int.Parse(Console.ReadLine());

if ((num % 2) != 0) throw new NotEvenException("Not an even number ");

else

Console.WriteLine("Its even number ");

}

catch (NotEvenException e) { Console.WriteLine(e.Message); }

}

static void Main(string[] args)

{

delegateExample();

handlingExceptions();

Console.ReadKey();

}

}

}

