**PRACTICAL NO.02**

**2. Working with Object Oriented C# and ASP .NET**

a. Create simple application to perform following operations

i. Finding factorial Value

ii. Money Conversion

iii. Quadratic Equation

iv. Temperature Conversion

**CODE:**

//Name:- Tukaram Manjarekar

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

//tukaram manjarekar

namespace pract2a

{

class Program

{

static void fact()

{

Console.Write("Enter a number to find factorial :");

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

if (n == 0 || n == 1)

n = 1;

long fact = 1;

for (int i=2;i<=n;i++)

{

fact=fact\*i;

}

Console.WriteLine($"factorial of {n} is :{fact}");

Console.ReadLine();

}

static void moneyconvert()

{

Console.Write("enter amount in USD : ");

decimal usd = Convert.ToDecimal(Console.ReadLine());

decimal convertedAmount = ConvertToINR(usd);

Console.WriteLine($"amount in INR :{convertedAmount}");

Console.ReadLine();

}

static decimal ConvertToINR(decimal usd)

{

decimal conversionRate = 79;

return usd \* conversionRate;

}

static void quadraticequation()

{

Console.WriteLine("enter coefficients (a,b,c) of the quadratic equation ax^2+bx+c=0 : ");

Console.Write("Enter a: ");

double a = Convert.ToDouble(Console.ReadLine());

Console.Write("Enter b: ");

double b = Convert.ToDouble(Console.ReadLine());

Console.Write("Enter c: ");

double c = Convert.ToDouble(Console.ReadLine());

double discriminant = b \* b - 4 \* a \* c;

if (discriminant > 0)

{

double root1 = (-b + Math.Sqrt(discriminant)) / (2 \* a);

double root2 = (-b + Math.Sqrt(discriminant)) / (2 \* a);

Console.WriteLine($"Roots are real and different.\nRoot1 = {root1},Root2 = {root2}");

}

else

{

double realPart = -b / (2 \* a);

double imagenarypart = Math.Sqrt(-discriminant) / (2 \* a);

Console.WriteLine($"Roots are complex \nRoot1 = {realPart} + {imagenarypart}i,Root2 = {realPart}-{imagenarypart}i");

}

Console.ReadLine();

}

static void tmp()

{

Console.Write("enter temperature in celsius : ");

double celsius = Convert.ToDouble(Console.ReadLine());

double fahrenheit = celsius \* 9 / 5 + 32;

double kelvin = celsius + 273.15;

Console.WriteLine($"temperature in fahrenheit :{fahrenheit}");

Console.WriteLine($"temperature in kelvin : {kelvin}");

Console.ReadLine();

}

static void Main(string[] args)

{

fact();

moneyconvert();

tmp();

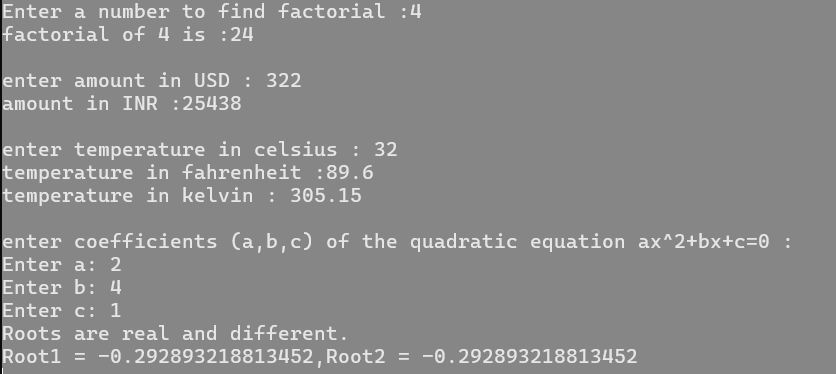
quadraticequation();

}

}

}

**OUTPUT:**



b. Create simple application to demonstrate use of following concepts

i. Function Overloading

ii. Inheritance (all types)

iii. Constructor overloading

**CODE:**

using System;

namespace ConceptsDemo

{

// Base class for inheritance demonstration

public class Animal

{

public string Name;

// Constructor overloading

public Animal()

{

Console.WriteLine("Animal created.");

}

public Animal(string name)

{

Name = name;

Console.WriteLine($"Animal created with name: {name}");

}

public void Speak()

{

Console.WriteLine("Animal speaks.");

}

}

// Single Inheritance: Dog class inherits from Animal class

public class Dog : Animal

{

public Dog() : base()

{

Console.WriteLine("Dog created.");

}

public Dog(string name) : base(name)

{

Console.WriteLine($"Dog created with name: {name}");

}

public void Bark()

{

Console.WriteLine("Dog barks.");

}

}

// Multilevel Inheritance: Puppy class inherits from Dog class

public class Puppy : Dog

{

public Puppy() : base()

{

Console.WriteLine("Puppy created.");

}

public Puppy(string name) : base(name)

{

Console.WriteLine($"Puppy created with name: {name}");

}

public void Play()

{

Console.WriteLine("Puppy plays.");

}

}

// Hierarchical Inheritance: Cat class also inherits from Animal class

public class Cat : Animal

{

public Cat() : base()

{

Console.WriteLine("Cat created.");

}

public Cat(string name) : base(name)

{

Console.WriteLine($"Cat created with name: {name}");

}

public void Meow()

{

Console.WriteLine("Cat meows.");

}

}

// Class demonstrating function overloading

public class Calculator

{

// Function overloading with different parameter counts

public int Add(int a, int b)

{

return a + b;

}

public int Add(int a, int b, int c)

{

return a + b + c;

}

// Function overloading with different parameter types

public double Add(double a, double b)

{

return a + b;

}

}

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Function Overloading Demo:");

Calculator calc = new Calculator();

Console.WriteLine($"Add(2, 3): {calc.Add(2, 3)}");

Console.WriteLine($"Add(2, 3, 4): {calc.Add(2, 3, 4)}");

Console.WriteLine($"Add(2.5, 3.5): {calc.Add(2.5, 3.5)}");

Console.WriteLine("\nSingle Inheritance and Constructor Overloading Demo:");

Dog dog = new Dog("Tommy");

dog.Speak();

dog.Bark();

Console.WriteLine("\nMultilevel Inheritance Demo:");

Puppy puppy = new Puppy("rokey");

puppy.Speak();

puppy.Bark();

puppy.Play();

Console.WriteLine("\nHierarchical Inheritance Demo:");

Cat cat = new Cat("sweety");

cat.Speak();

cat.Meow();

Console.WriteLine("\nConstructor Overloading Demo:");

Animal animal1 = new Animal();

Animal animal2 = new Animal("tiger");

Console.ReadLine();

}

}

}

**OUTPUT:**

