**Best Programming Practices in C#**

1. Use meaningful variable and method names.
2. Follow PascalCase for class names and method names.
3. Follow camelCase for variable names.
4. Properly structure code with clear comments and modular methods.

**Sample Program 1:** Create a program to find the sum of all the digits of a number given by a user using an array and display the sum.

1. Use Math.random() and get a 4-digit random integer number
2. Write a method to count digits in the number
3. Write a method to return an array of digits from a given number.
4. Write a method to Find the sum of the digits of the number in the array
5. Finally, display the sum of the digits of the number

using System;

class SumOfDigits

{

// Generate a 4-digit random number

public int Get4DigitRandomNumber()

{

Random random = new Random();

return random.Next(1000, 10000); // Generates a random number between 1000 and 9999

}

// Count the number of digits in a given number

public int CountDigits(int number)

{

int count = 0;

while (number > 0)

{

count++;

number /= 10;

}

return count;

}

// Extract digits from the number and store them in an array

public int[] GetDigits(int number, int count)

{

int[] digits = new int[count];

for (int i = count - 1; i >= 0; i--)

{

digits[i] = number % 10;

number /= 10;

}

return digits;

}

// Calculate the sum of the digits in the array

public int SumArray(int[] array)

{

int sum = 0;

foreach (int digit in array)

{

sum += digit;

}

return sum;

}

static void Main(string[] args)

{

// Instantiate the SumOfDigits class

SumOfDigits sumOfDigits = new SumOfDigits();

// Generate a 4-digit random number

int number = sumOfDigits.Get4DigitRandomNumber();

Console.WriteLine("The Random Number is: " + number);

// Count the number of digits

int count = sumOfDigits.CountDigits(number);

Console.WriteLine("The Count of Digits is: " + count);

// Extract digits into an array

int[] digits = sumOfDigits.GetDigits(number, count);

// Calculate the sum of the digits

int sum = sumOfDigits.SumArray(digits);

// Display the sum of the digits

Console.WriteLine("Sum of Digits: " + sum);

}

}



# Level 2 Practice Programs

1.Create a program to find the factors of a number taken as user input, store the factors in an array and display the factors. Also find the sum, sum of square of factors and product of the factors and display the results

**Hint =>**

1. Take the input for a number
2. Write a ***static*** Method to find the factors of the number and save them in an array and return the array.
3. To find factors and save to array will have two loops. The first loop to find the count and initialize the array with the count. And the second loop save the factors into the array
4. Write a method to find the sum of the factors using factors array
5. Write a method to find the product of the factors using factors array
6. Write a method to find the sum of square of the factors using ***Math.Pow()*** method

| using System;  class Solution {  static int [] findFactors(int number){  int count = 0;   // First loop to count the number of factors  for (int i = 1; i <= number; i++) {  if (number % i == 0)  count++;  }   // Initialize the array with the count of factors  int[] factors = new int[count];  int index = 0;   for (int i = 1; i <= number; i++) {  if (number % i == 0) {  factors[index] = i;  index++;  }  }  return factors;  }   // method to calculate the sum  public static int CalculateSum(int[] factors) {  int sum = 0;  foreach (int factor in factors) {  sum += factor;  }  return sum;  }   //method to calculate the product  public static long CalculateProduct(int[] factors) {  long product = 1;  foreach (int factor in factors) {  product \*= factor;  }  return product;  }   // Method to calculate the sum of the square of factors  public static int CalculateSumOfSquares(int[] factors) {  int sumOfSquares = 0;  foreach (int factor in factors) {  sumOfSquares += (int)Math.Pow(factor, 2);  }  return sumOfSquares;  }   public static void Main(){  // prompt the user to enter a number  Console.Write("Enter a number: ");  int num = int.Parse(Console.ReadLine());    int[] factors = findFactors(num);    Console.WriteLine("Factors of the number: ");  Console.WriteLine(string.Join(", ", factors));   // Calculate and display the sum of the factors  int sum = CalculateSum(factors);  Console.WriteLine("Sum of the factors: "+sum);   // Calculate and display the product of the factors  long product = CalculateProduct(factors);  Console.WriteLine("Product of the factors: "+product);   // Calculate and display the sum of the square of the factors  int sumOfSquares = CalculateSumOfSquares(factors);  Console.WriteLine("Sum of the square of the factors: "+sumOfSquares);  } } |
| --- |

2. Write a program to find the sum of n natural numbers using recursive method and compare the result with the formulae n\*(n+1)/2 and show the result from both computations is correct.

**Hint =>**

a. Take the user input number and check whether it's a Natural number, if not exit

b. Write a Method to find the sum of n natural numbers using **recursion**

c. Write a Method to find the sum of n natural numbers using the formulae

n\*(n+1)/2

d. Compare the two results and print the result

| using System;  class Solution{  // method for the recursive sum  public static int recursiveSum(int n){  if (n==1){  return 1;  }  return (n+recursiveSum(n-1));  }  // method for the formal sum  public static int formalSum(int n){  return n\*(n+1)/2;  }  static void Main(string[] args){  // Prompt the user to enter a number  Console.Write("Enter a number: ");  int n = int.Parse(Console.ReadLine());    Console.WriteLine("The Recursive sum of the numbers "+recursiveSum(n));  Console.WriteLine("The formal sum of the numbers "+formalSum(n));    // output the compare of both the results  if(recursiveSum(n)==formalSum(n)){  Console.WriteLine("Both results are equal");  }  else{  Console.WriteLine("Both are not equal");  }  } } |
| --- |

3. Write a program that takes a year as input and outputs the Year is a Leap Year or not

**Hint =>**

a. The LeapYear program only works for year >= 1582, corresponding to a year in the

Gregorian calendar.

b. Also Leap year is divisible by 4 and not divisible by 100 or divisible by 400

c. Write a method to check for Leap Year using the conditions a and b

| using System;  class Solution {  // Method to check if a year is a leap year  public static bool IsLeapYear(int year) {  if (year < 1582) {  Console.WriteLine("The year must be 1582 or later");  return false;  }   // Check leap year conditions  if ((year % 4 == 0 && year % 100 != 0) || year % 400 == 0) {  return true; // Leap year  }  return false; // Not a leap year  }   public static void Main() {  // Input: Take year input from the user  Console.Write("Enter a year (>= 1582): ");  int year = int.Parse(Console.ReadLine());     // Check if the year is a leap year  bool isLeap = IsLeapYear(year);   // Output the result  if (isLeap) {  Console.WriteLine("This is a Leap Year.");  }  else{  Console.WriteLine("This is NOT a Leap Year.");  }  } } |
| --- |

**4.**  Extend or Create a ***UnitConvertor*** utility class similar to the one shown in the notes to do the following. Please define ***static*** methods for all the UnitConvertor class methods. E.g.

***public static double ConvertKmToMiles(double km)* =>**

a. Method To convert kilometers to miles and return the value. Use the following code double km2miles = 0.621371;

b. Method to convert miles to kilometers and return the value. Use the following code double miles2km = 1.60934;

c. Method to convert meters to feet and return the value. Use the following code to convert double meters2feet = 3.28084;

d. Method to convert feet to meters and return the value. Use the following code to convert double feet2meters = 0.3048;

| using System;  class UnitConverter {  // Method to convert kilometers to miles  public static double ConvertKmToMiles(double km) {  double km2miles = 0.621371;  return km \* km2miles;  }   // Method to convert miles to kilometers  public static double ConvertMilesToKm(double miles) {  double miles2km = 1.60934;  return miles \* miles2km;  }   // Method to convert meters to feet  public static double ConvertMetersToFeet(double meters) {  double meters2feet = 3.28084;  return meters \* meters2feet;  }   // Method to convert feet to meters  public static double ConvertFeetToMeters(double feet) {  double feet2meters = 0.3048;  return feet \* feet2meters;  } }  class Solution {  public static void Main() {  // Demonstrate conversions  Console.WriteLine("Calculator for conversions");    // Kilometers to Miles  Console.Write("Enter kilometers to convert to miles: ");  double km = double.Parse(Console.ReadLine());  Console.WriteLine(km + " km is " + UnitConverter.ConvertKmToMiles(km).ToString("F2") + " miles.");    // Miles to Kilometers  Console.Write("Enter miles to convert to kilometers: ");  double miles = double.Parse(Console.ReadLine());  Console.WriteLine(miles + " miles is " + UnitConverter.ConvertMilesToKm(miles).ToString("F2") + " kilometers.");    // Meters to Feet  Console.Write("Enter meters to convert to feet: ");  double meters = double.Parse(Console.ReadLine());  Console.WriteLine(meters + " meters is " + UnitConverter.ConvertMetersToFeet(meters).ToString("F2") + " feet.");    // Feet to Meters  Console.Write("Enter feet to convert to meters: ");  double feet = double.Parse(Console.ReadLine());  Console.WriteLine(feet + " feet is " + UnitConverter.ConvertFeetToMeters(feet).ToString("F2") + " meters.");  } } |
| --- |

5. Extend or Create a ***UnitConvertor*** utility class similar to the one shown in the notes to do the following. Please define ***static*** methods for all the UnitConvertor class methods. E.g.

***public static double ConvertYardsToFeet(double yards)* =>**

a. Method to convert yards to feet and return the value. Use following code to convert double yards2feet = 3;

b. Method to convert feet to yards and return the value. Use following code to convert double feet2yards = 0.333333;

c. Method to convert meters to inches and return the value. Use following code to convert double meters2inches = 39.3701;

d. Method to convert inches to meters and return the value. Use following code to convert double inches2meters = 0.0254;

e. Method to convert inches to centimeters and return the value. Use the following code double inches2cm = 2.54;

| using System;  class UnitConverter {  // Method to convert yards to feet  public static double ConvertYardsToFeet(double yards) {  double yards2feet = 3;  return yards \* yards2feet;  }   // Method to convert feet to yards  public static double ConvertFeetToYards(double feet) {  double feet2yards = 0.333333;  return feet \* feet2yards;  }   // Method to convert meters to inches  public static double ConvertMetersToInches(double meters) {  double meters2inches = 39.3701;  return meters \* meters2inches;  }   // Method to convert inches to meters  public static double ConvertInchesToMeters(double inches) {  double inches2meters = 0.0254;  return inches \* inches2meters;  }   // Method to convert inches to centimeters  public static double ConvertInchesToCentimeters(double inches) {  double inches2cm = 2.54;  return inches \* inches2cm;  } }  class Solution {  static void Main(string[] args) {  Console.WriteLine("Calculator for conversions");   // Yards to Feet  Console.Write("Enter yards to convert to feet: ");  double yards = double.Parse(Console.ReadLine());  Console.WriteLine(yards + " yards is " + UnitConverter.ConvertYardsToFeet(yards).ToString("F2") + " feet.");   // Feet to Yards  Console.Write("Enter feet to convert to yards: ");  double feet = double.Parse(Console.ReadLine());  Console.WriteLine(feet + " feet is " + UnitConverter.ConvertFeetToYards(feet).ToString("F2") + " yards.");   // Meters to Inches  Console.Write("Enter meters to convert to inches: ");  double meters = double.Parse(Console.ReadLine());  Console.WriteLine(meters + " meters is " + UnitConverter.ConvertMetersToInches(meters).ToString("F2") + " inches.");   // Inches to Meters  Console.Write("Enter inches to convert to meters: ");  double inches = double.Parse(Console.ReadLine());  Console.WriteLine(inches + " inches is " + UnitConverter.ConvertInchesToMeters(inches).ToString("F2") + " meters.");   // Inches to Centimeters  Console.Write("Enter inches to convert to centimeters: ");  inches = double.Parse(Console.ReadLine());  Console.WriteLine(inches + " inches is " + UnitConverter.ConvertInchesToCentimeters(inches).ToString("F2") + " centimeters.");  } } |
| --- |

6. Extend or Create a ***UnitConvertor*** utility class similar to the one shown in the notes to do the following. Please define ***static*** methods for all the UnitConvertor class methods. E.g.

***public static double convertFarhenheitToCelsius(double farhenheit)* =>**

a. Method to convert Fahrenheit to Celsius and return the value. Use the following code double farhenheit2celsius = (farhenheit - 32) \* 5 / 9;

b. Method to convert Celsius to Fahrenheit and return the value. Use the following code double celsius2farhenheit = (celsius \* 9 / 5) + 32;

c. Method to convert pounds to kilograms and return the value. Use the following code double pounds2kilograms = 0.453592;

d. Method to convert kilograms to pounds and return the value. Use the following code double kilograms2pounds = 2.20462;

e. Method to convert gallons to liters and return the value. Use following code to convert double gallons2liters = 3.78541;

f. Method to convert liters to gallons and return the value. Use following code to convert double liters2gallons = 0.264172;

| using System;  class UnitConverter {  // Method to convert Fahrenheit to Celsius  public static double ConvertFahrenheitToCelsius(double fahrenheit) {  return (fahrenheit - 32) \* 5 / 9;  }   // Method to convert Celsius to Fahrenheit  public static double ConvertCelsiusToFahrenheit(double celsius) {  return (celsius \* 9 / 5) + 32;  }   // Method to convert pounds to kilograms  public static double ConvertPoundsToKilograms(double pounds) {  double pounds2kilograms = 0.453592;  return pounds \* pounds2kilograms;  }   // Method to convert kilograms to pounds  public static double ConvertKilogramsToPounds(double kilograms) {  double kilograms2pounds = 2.20462;  return kilograms \* kilograms2pounds;  }   // Method to convert gallons to liters  public static double ConvertGallonsToLiters(double gallons) {  double gallons2liters = 3.78541;  return gallons \* gallons2liters;  }   // Method to convert liters to gallons  public static double ConvertLitersToGallons(double liters) {  double liters2gallons = 0.264172;  return liters \* liters2gallons;  } }  class Solution {  public static void Main() {  Console.WriteLine("Calculator for conversions");   // Fahrenheit to Celsius  Console.Write("Enter temperature in Fahrenheit to convert to Celsius: ");  double fahrenheit = double.Parse(Console.ReadLine());  Console.WriteLine(fahrenheit + "°F is " + UnitConverter.ConvertFahrenheitToCelsius(fahrenheit).ToString("F2") + "°C.");   // Celsius to Fahrenheit  Console.Write("Enter temperature in Celsius to convert to Fahrenheit: ");  double celsius = double.Parse(Console.ReadLine());  Console.WriteLine(celsius + "°C is " + UnitConverter.ConvertCelsiusToFahrenheit(celsius).ToString("F2") + "°F.");   // Pounds to Kilograms  Console.Write("Enter weight in pounds to convert to kilograms: ");  double pounds = double.Parse(Console.ReadLine());  Console.WriteLine(pounds + " pounds is " + UnitConverter.ConvertPoundsToKilograms(pounds).ToString("F2") + " kilograms.");   // Kilograms to Pounds  Console.Write("Enter weight in kilograms to convert to pounds: ");  double kilograms = double.Parse(Console.ReadLine());  Console.WriteLine(kilograms + " kilograms is " + UnitConverter.ConvertKilogramsToPounds(kilograms).ToString("F2") + " pounds.");   // Gallons to Liters  Console.Write("Enter volume in gallons to convert to liters: ");  double gallons = double.Parse(Console.ReadLine());  Console.WriteLine(gallons + " gallons is " + UnitConverter.ConvertGallonsToLiters(gallons).ToString("F2") + " liters.");   // Liters to Gallons  Console.Write("Enter volume in liters to convert to gallons: ");  double liters = double.Parse(Console.ReadLine());  Console.WriteLine(liters + " liters is " + U nitConverter.ConvertLitersToGallons(liters).ToString("F2") + " gallons.");  } } |
| --- |

7. Write a program to take user input for the age of all 10 students in a class and check whether the student can vote depending on his/her age is greater or equal to 18.

**Hint =>**

a. Create a class ***public class StudentVoteChecker*** and define a method ***public boolean CanStudentVote(int age)*** which takes in age as a parameter and returns true or false

b. Inside the method firstly validate the age for a negative number, if a negative return is false cannot vote. For valid age check for age is 18 or above return true; else return false;

c. In the main function define an array of 10 integer elements, loop through the array by take user input for the student's age, call **CanStudentVote()** and display the result

| using System;  public class StudentVoteChecker {  // Method to check if the student can vote  public static bool CanStudentVote(int age) {  if (age < 0) {  return false;  }  else if (age >= 18) {  return true;  }  else {  return false;  }  } }  class Solution{  static void Main() {  // Array to hold ages of 10 students  int[] studentAges = new int[10];   Console.WriteLine("Enter the ages of 10 students:");   // Loop to get user input for student ages  for (int i = 0; i < studentAges.Length; i++) {  Console.Write("Enter age of student " + (i + 1) + ": ");  studentAges[i] = int.Parse(Console.ReadLine());  }   Console.WriteLine("\nVoting Eligibility Results:");   // Loop to check voting eligibility for each student  for (int i = 0; i < studentAges.Length; i++) {  bool canVote = StudentVoteChecker.CanStudentVote(studentAges[i]);   if (studentAges[i] < 0) {  Console.WriteLine("Student " + (i + 1) + ": Invalid age (" + studentAges[i] + ").");  }  else if (canVote) {  Console.WriteLine("Student " + (i + 1) + ": Age " + studentAges[i] + " - Eligible to vote.");  }  else {  Console.WriteLine("Student " + (i + 1) + ": Age " + studentAges[i] + " - Not eligible to vote.");  }  }  } } |
| --- |

8. Create a program to find the youngest friends among 3 Amar, Akbar and Anthony based on their ages and tallest among the friends based on their heights and display it

**Hint =>**

a. Take user input for age and height for the 3 friends and store it in two arrays each to store the values for age and height of the 3 friends

b. Write a Method to find the youngest of the 3 friends

c. Write a Method to find the tallest of the 3 friends

| using System;  public class FriendsDetails {  // Method to find the youngest friend  public static int FindYoungest(int[] ages) {  int youngestIndex = 0;   for (int i = 1; i < ages.Length; i++) {  if (ages[i] < ages[youngestIndex]) {  youngestIndex = i;  }  }  return youngestIndex;  }   // Method to find the tallest friend  public static int FindTallest(float[] heights) {  int tallestIndex = 0;   for (int i = 1; i < heights.Length; i++) {  if (heights[i] > heights[tallestIndex]) {  tallestIndex = i;  }  }  return tallestIndex;  } }  class Solution {  static void Main() {  // Arrays to store ages and heights of the 3 friends  int[] ages = new int[3];  float[] heights = new float[3];   string[] friends = { "Amar", "Akbar", "Anthony" };   Console.WriteLine("Enter the age and height for the friends:");   // Loop to take user input for ages and heights  for (int i = 0; i < friends.Length; i++) {  Console.Write("Enter age of " + friends[i] + ": ");  ages[i] = int.Parse(Console.ReadLine());   Console.Write("Enter height (in cm) of " + friends[i] + ": ");  heights[i] = float.Parse(Console.ReadLine());  }   // Find the youngest friend  int youngestIndex = FriendsDetails.FindYoungest(ages);   // Find the tallest friend  int tallestIndex = FriendsDetails.FindTallest(heights);   // Display the results  Console.WriteLine("Youngest Friend: " + friends[youngestIndex] + " (Age: " + ages[youngestIndex] + ")");  Console.WriteLine("Tallest Friend: " + friends[tallestIndex] + " (Height: " + heights[tallestIndex] + " cm)");  } } |
| --- |

9. Write a program to take user input for 5 numbers and check whether a number is positive or negative. Further for positive numbers check if the number is even or odd. Finally compare the first and last elements of the array and display if they are equal, greater, or less

**Hint =>**

a. Write a Method to Check whether the number is positive or negative

b. Write a Method to check whether the number is even or odd

c. Write a Method to compare two numbers and return 1 if number1 > number2 or 0 if both are equal or -1 if number1 < number2

d. In the main program, Loop through the array using the length call the method ***isPositive()*** and if positive call method ***isEven()*** and print accordingly

e. If the number is negative, print negative.

f. Finally compare the first and last element of the array by calling the method ***compare()*** and display if they are equal, greater, or less

| using System;  public class NumberChecker {  // Method to check if a number is positive or negative  public static bool IsPositive(int number) {  return number >= 0;  }   // Method to check if a number is even or odd  public static bool IsEven(int number) {  return number % 2 == 0;  }   // Method to compare two numbers  public static int Compare(int number1, int number2) {  if (number1 > number2) return 1;  if (number1 == number2) return 0;  return -1;  } }  class Solution {  static void Main() {  int[] numbers = new int[5];   Console.WriteLine("Enter 5 numbers:");   // Taking user input for the array  for (int i = 0; i < numbers.Length; i++) {  Console.Write("Enter number " + (i + 1) + ": ");  numbers[i] = int.Parse(Console.ReadLine());  }   Console.WriteLine("\nAnalysis of the numbers:");   // Loop through the array to analyze each number  for (int i = 0; i < numbers.Length; i++) {  if (NumberChecker.IsPositive(numbers[i])) {  if (NumberChecker.IsEven(numbers[i])) {  Console.WriteLine("Number " + numbers[i] + " is Positive and Even.");  }  else {  Console.WriteLine("Number " + numbers[i] + " is Positive and Odd.");  }  }  else {  Console.WriteLine("Number " + numbers[i] + " is Negative.");  }  }    // Comparing the first and last elements of the array  int comparisonResult = NumberChecker.Compare(numbers[0], numbers[numbers.Length - 1]);  Console.WriteLine("\nComparison between first and last elements:");   if (comparisonResult == 1) {  Console.WriteLine("The first element (" + numbers[0] + ") is greater than the last element (" + numbers[numbers.Length - 1] + ").");  }  else if (comparisonResult == 0) {  Console.WriteLine("The first element (" + numbers[0] + ") is equal to the last element (" + numbers[numbers.Length - 1] + ").");  }  else {  Console.WriteLine("The first element (" + numbers[0] + ") is less than the last element (" + numbers[numbers.Length - 1] + ").");  }  } } |
| --- |

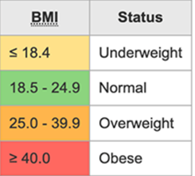
10. An organization took up the exercise to find the Body Mass Index (BMI) of all the persons in the team of 10 members. For this create a program to find the BMI and display the height, weight, BMI and status of each individual

**Hint =>**

a. Take user input in double for the weight (in kg) of the person and height (in cm) for the person and store it in the corresponding 2D array of 10 rows and 3 columns. The First Column storing the weight, the second column storing the height in cm and the third column is the BMI

b. Create a Method to find the BMI of every person and populate the array. Use the formula BMI = weight / (height \* height). Note unit is kg/m^2. For this convert cm to meter

c. Create a Method to determine the BMI status using the logic shown in the figure below. and return the array of all the persons BMI Status.



| using System;  class BMI\_Calculator {  // Method to calculate BMI and populate the array  public static void CalculateBMI(double[,] data) {  for (int i = 0; i < 10; i++) {  double weight = data[i, 0];  double heightInMeters = data[i, 1] / 100;  double bmi = weight / (heightInMeters \* heightInMeters);  data[i, 2] = bmi;  }  }   // Method to determine BMI status  public static string[] DetermineBMIStatus(double[,] data) {  string[] status = new string[10];   for (int i = 0; i < 10; i++) {  double bmi = data[i, 2];   if (bmi <= 18.4) {  status[i] = "Underweight";  }  else if (bmi >= 18.5 && bmi < 24.9) {  status[i] = "Normal weight";  }  else if (bmi >= 25 && bmi < 39.9) {  status[i] = "Overweight";  }  else {  status[i] = "Obese";  }  }  return status;  } }  class Solution {  static void Main() {  double[,] data = new double[10, 3];  string[] status = new string[10];   // Taking user input for weight and height  Console.WriteLine("Enter the weight (in kg) and height (in cm) for 10 individuals:");   for (int i = 0; i < 10; i++) {  Console.WriteLine("\nPerson " + (i + 1) + ":");  Console.Write("Enter weight (in kg): ");  data[i, 0] = double.Parse(Console.ReadLine());  Console.Write("Enter height (in cm): ");  data[i, 1] = double.Parse(Console.ReadLine());  }   // Calculate BMI for all individuals  BMI\_Calculator.CalculateBMI(data);   // Get BMI status  status = BMI\_Calculator.DetermineBMIStatus(data);   // Display results  Console.WriteLine("BMI and Status for each individual:");   for (int i = 0; i < 10; i++) {  Console.WriteLine("Person " + (i + 1) + ":");  Console.WriteLine("Weight: " + data[i, 0] + " kg");  Console.WriteLine("Height: " + data[i, 1] + " cm");  Console.WriteLine("BMI: " + data[i, 2].ToString("0.00"));  Console.WriteLine("Status: " + status[i]);  }  } } |
| --- |

11. Write a program Quadratic to find the roots of the equation ax2+ bx + c. Use Math functions ***Math.pow()*** and ***Math.sqrt()***

**Hint =>**

a. Take a, b, and c as input values to find the roots of x.

b. The roots are computed using the following formulae

delta = b2+ 4\*a\*c

If delta is positive the find the two roots using formulae

root1 of x = (-b + delta)/(2\*a)

root1 of x = (-b - delta)/(2\*a)

If delta is zero then there is only one root of x

root of x = -b/(2\*a)

If delta is negative return empty array or nothing

c. Write a Method to find find the roots of a quadratic equation and return the roots

| using System;  class Solution {  // Method to find the roots of the quadratic equation  public static double[] FindRoots(double a, double b, double c) {  double delta = Math.Pow(b, 2) - 4 \* a \* c;   if (delta > 0) {  double root1 = (-b + Math.Sqrt(delta)) / (2 \* a);  double root2 = (-b - Math.Sqrt(delta)) / (2 \* a);  return new double[] { root1, root2 };  }  else if (delta == 0) {  double root = -b / (2 \* a);  return new double[] { root };  }  else {  return new double[] { };  }  }   static void Main() {  // Taking input values for a, b, and c  Console.Write("Enter the coefficient a: ");  double a = double.Parse(Console.ReadLine());   Console.Write("Enter the coefficient b: ");  double b = double.Parse(Console.ReadLine());   Console.Write("Enter the coefficient c: ");  double c = double.Parse(Console.ReadLine());   // Finding the roots  double[] roots = FindRoots(a, b, c);   // Displaying the roots  if (roots.Length == 2) {  Console.WriteLine("The two roots are: root1 = " + roots[0] + ", root2 = " + roots[1]);  }  else if (roots.Length == 1) {  Console.WriteLine("The only root is: root = " + roots[0]);  }  else {  Console.WriteLine("The equation has no real roots.");  }  } } |
| --- |

12. Write a program that generates five 4 digit random values and then finds their average value, and their minimum and maximum value. Use Math.Random(), Math.Min(), and Math.Max().

**Hint =>**

a. Write a method that generates array of 4 digit random numbers given the size as a parameter as shown in the method signature

***public int[] Generate4DigitRandomArray(int size)***

b. Write a method to find average, min and max value of an array

***public double[] FindAverageMinMax(int[] numbers)***

| **using System;  class Solution{  // Method to generate an array of 4-digit random numbers  public static int[] Generate4DigitRandomArray(int size) {  Random rand = new Random();  int[] numbers = new int[size];   for (int i = 0; i < size; i++) {  // Generate a random 4-digit number (1000 to 9999)  numbers[i] = rand.Next(1000, 10000);  }  return numbers;  }   // Method to find the average, minimum, and maximum value of an array  public static double[] FindAverageMinMax(int[] numbers) {  double sum = 0;  int min = numbers[0];  int max = numbers[0];   for (int i = 0; i < numbers.Length; i++) {  sum += numbers[i];  min = Math.Min(min, numbers[i]);  max = Math.Max(max, numbers[i]);  }   double average = sum / numbers.Length;  return new double[] { average, min, max };  }   static void Main() {  // Generate an array of 5 random 4-digit numbers  int[] randomNumbers = Generate4DigitRandomArray(5);   // Find the average, minimum, and maximum values  double[] results = FindAverageMinMax(randomNumbers);   // Display the generated random numbers  Console.WriteLine("Generated 4-digit random numbers:");  foreach (int number in randomNumbers)  {  Console.WriteLine(number);  }   // Display the results (average, min, max)  Console.WriteLine("Average: " + results[0].ToString("0.00"));  Console.WriteLine("Minimum: " + results[1]);  Console.WriteLine("Maximum: " + results[2]);  } }** |
| --- |