**C#** **ASSINGMENTS**

**Assignment1:**

**1 Write a C# program to find the sum of all elements in an integer array using a loop.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment1

{

internal class Program

{

static void Main(string[] args)

{

int[] arr = new int[5];

int sum = 0;

Console.WriteLine("Enter 5 Elements");

for (int i = 0; i < arr.Length; i++)

{

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

}

for (int i = 0; i < arr.Length; i++)

{

sum += arr[i];

}

Console.WriteLine("Sum is :" + sum);

}

}

}

**2 Create a C# program that calculates the average of values in a floating-point array using a loop.**

namespace Assignment1

{

internal class Program

{

static void Main(string[] args)

{

int[] arr = new int[5];

double sum = 0;

Console.WriteLine("Enter 5 Elements");

for (int i = 0; i < arr.Length; i++)

{

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

}

for (int i = 0; i < arr.Length; i++)

{

sum += arr[i];

}

double avg = sum / 5;

Console.WriteLine("Average is :" + avg);

}

}

}

**3 Develop a C# program that finds the largest element in an integer array using a loop and if-else statements.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment1

{

internal class Program

{

static void Main(string[] args)

{

int[] arr = new int[5];

int max = 0;

Console.WriteLine("Enter 5 Elements");

for (int i = 0; i < arr.Length; i++)

{

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

}

for (int i = 0; i < arr.Length; i++)

{

if (arr[i] > max)

{

max = arr[i];

}

}

Console.WriteLine(" Maximum Element is :" + max);

}

}

}

**4 Write a C# program that counts the number of even and odd elements in an integer array using a loop and if-else statements.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment1

{

internal class Program

{

static void Main(string[] args)

{

int[] arr = new int[5];

int even = 0;

int odd = 0;

Console.WriteLine("Enter 5 Elements");

for (int i = 0; i < arr.Length; i++)

{

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

}

for (int i = 0; i < arr.Length; i++)

{

if (arr[i] % 2 == 0)

{

even++;

}

else

{

odd++;

}

}

Console.WriteLine(" Total Even Element is :" + even);

Console.WriteLine(" Total Odd Element is :" + odd);

}

}

}

**5 Implement a C# program that reverses the elements of an integer array using a loop.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment1

{

internal class Program

{

static void Main(string[] args)

{

int[] arr = new int[5];

int start = 0;

int end = arr.Length - 1;

Console.WriteLine("Enter 5 Elements");

for (int i = 0; i < arr.Length; i++)

{

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

}

while (start < end)

{

int temp = arr[start];

arr[start] = arr[end];

arr[end] = temp;

start++;

end--;

}

Console.WriteLine("Reverse Array is :");

for (int i = 0; i < arr.Length; i++)

{

Console.Write(arr[i] + " ");

}

Console.ReadLine();

}

}

}

**6 Create a C# program that multiplies each element in an integer array by a specified factor using a loop.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment1

{

internal class Program

{

static void Main(string[] args)

{

int[] arr = new int[5];

Console.WriteLine("Enter 5 Elements");

for (int i = 0; i < arr.Length; i++)

{

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

arr[i] = arr[i] \* 2;

}

Console.WriteLine("Array is :");

for (int i = 0; i < arr.Length; i++)

{

Console.Write(arr[i] + " ");

}

Console.ReadLine();

}

}

}

**7 Write a C# program that searches for a specific value in an integer array using a loop and returns its index if found.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment1

{

internal class Program

{

static void Main(string[] args)

{

int[] arr = new int[5];

Console.WriteLine("Enter 5 Elements");

for (int i = 0; i < arr.Length; i++)

{

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

}

Console.WriteLine("Enter Elements to be search");

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

for (int i = 0; i < arr.Length; i++)

{

if (arr[i] == n)

{

Console.WriteLine("Element found at index :" + i);

Console.ReadLine();

return;

}

}

Console.WriteLine("Element not found");

Console.ReadLine();

}

}

}

**8 Develop a C# program that finds the second smallest element in an integer array using loops and sorting techniques.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment1

{

internal class Program

{

static void Main(string[] args)

{

int[] arr = new int[5];

Console.WriteLine("Enter 5 Elements");

for (int i = 0; i < arr.Length; i++)

{

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

}

for (int i = 0; i < arr.Length; i++)

{

for (int j = 1; j < arr.Length; j++)

{

if (arr[j - 1] > arr[j])

{

int temp = arr[j];

arr[j] = arr[j - 1];

arr[j - 1] = temp;

}

}

}

Console.WriteLine("Array is:");

for (int i = 0; i < arr.Length; i++)

{

Console.Write(arr[i] + " ");

}

Console.WriteLine();

Console.WriteLine("Second Minimum is: " + arr[1]);

Console.ReadLine();

}

}

}

**9 Create a C# program that removes all duplicates from an integer array using loops and additional data structures.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment1

{

internal class Program

{

static void Main(string[] args)

{

int[] arr = { 1, 1, 3, 3, 3, 4, 5, 5 };

int l = arr.Length;

int r = removeDuplicateFromArray(arr, l);

Console.WriteLine("Final Array is:");

for (int i = 0; i < r; i++)

{

Console.Write(arr[i] + " ");

}

Console.ReadLine();

}

static int removeDuplicateFromArray(int[] arr, int n)

{

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

return 0;

int[] arr2 = new int[n];

int j = 0;

for (int i = 0; i < n - 1; i++)

{

if (arr[i] != arr[i + 1])

{

arr2[j++] = arr[i];

}

}

arr2[j++] = arr[n - 1];

for (int i = 0; i < j; i++)

{

arr[i] = arr2[i];

}

return j;

}

}

}

**10 Write a C# program that finds the common elements between two integer arraysÂ usingÂ loops.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment1

{

internal class Program

{

static void Main(string[] args)

{

int[] arr = new int[5];

int[] arr2 = new int[5];

int j = 0;

int[] arr3 = new int[5];

Console.WriteLine("Enter 5 Elements of First Array");

for (int i = 0; i < arr.Length; i++)

{

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

}

Console.WriteLine("Enter 5 Elements of second Array");

for (int i = 0; i < arr2.Length; i++)

{

arr2[i] = Convert.ToInt32(Console.ReadLine());

}

for (int i = 0; i < arr.Length; i++)

{

for (int k = 0; k < arr2.Length; k++)

{

if (arr[i] == arr2[k])

{

arr3[j] = arr[i];

j++;

}

}

}

Console.WriteLine("Final Array with Common Elements are:");

for (int i = 0; i < j; i++)

{

Console.Write(arr3[i] + " ");

}

}

}

}

Assignment 2:

**1- Create a class calledEmployeewith properties for name, age, and salary. Implement a method to display employee details.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment2

{

internal class Program

{

static void Main(string[] args)

{

Employee e1 = new Employee(1, "Ritik", 123.56);

e1.displayData();

}

}

class Employee

{

public int id;

public string name;

public double salary;

public Employee(int id, string name, double salary)

{

this.id = id;

this.name = name;

this.salary = salary;

}

public void displayData()

{

Console.WriteLine("id :" + this.id);

Console.WriteLine("Name :" + this.name);

Console.WriteLine("Salary :" + this.salary);

Console.ReadKey();

}

}

}

**2 - Create a class called BankAccount with properties for account number, account holder name, and balance. Implement methods for deposit, withdrawal, and displaying the account details.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment2

{

internal class Program

{

static void Main(string[] args)

{

Bank b1 = new Bank(12345, "Ritik", 1000);

b1.displayData();

}

}

class Bank

{

public int AccountNumber;

public string name;

public double balance;

public Bank(int AccountNumber, string name, double balance)

{

this.AccountNumber = AccountNumber;

this.name = name;

this.balance = balance;

}

public void deposit(double balance)

{

balance += balance;

}

public void withdrawl(double amount)

{

if (amount > balance)

{

Console.WriteLine("Insufficinent Balance");

}

else

{

balance = balance - amount;

}

}

public void displayData()

{

Console.WriteLine("AccountNumber:" + this.AccountNumber);

Console.WriteLine("Name :" + this.name);

Console.WriteLine("Balance :" + this.balance);

Console.ReadKey();

}

}

}

**3- Create a static utility class named MathHelper with a static method CalculateAverage that takes an array of integers as input and returns their average.**

using System;

using System.Collections.Generic;

using System.Diagnostics.CodeAnalysis;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment2

{

internal class Program

{

static void Main(string[] args)

{

float[] nums = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

MathHelper.average(nums);

}

}

public static class MathHelper

{

static float sum= 0;

static float avg = 0;

public static void average(float[] nums)

{

for(int i=0;i<nums.Length; i++)

{

sum += nums[i];

}

avg = sum/nums.Length;

Console.WriteLine("Average is " + avg);

Console.ReadKey();

}

}

}

**4 - Implement a static logger class called Logger that has a method LogMessage for writing messages on console. Demonstrate its usage in a simple console application.**

using System;

using System.Collections.Generic;

using System.Diagnostics.CodeAnalysis;

using System.Linq;

using System.Runtime.CompilerServices;

using System.Text;

using System.Threading.Tasks;

namespace Assignment2

{

internal class Program

{

static void Main(string[] args)

{

Logger.setNameAndPassword("Ritik", 1234);

Logger.Login("Ritik", 1234);

}

}

public static class Logger

{

static string name;

static int password;

public static void setNameAndPassword(string username ,int userpassword)

{

name = username;

password = userpassword;

}

public static void Login(string username ,int userpassword)

{

if (username == name && userpassword == password)

{

Console.WriteLine("Login SuccessFull");

Console.ReadKey();

}

else

{

Console.WriteLine("Login Failed");

Console.ReadKey();

}

}

}

}

**5- Define a partial class Person with one part containing properties like FirstName and LastName, and another part with methods like PrintFullName to display the full name. Implement these parts in separate files.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment2

{

public partial class Person

{

public string firstName;

public string lastName;

public Person(string firstName, string lastName)

{

this.firstName = firstName;

this.lastName = lastName;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment2

{

public partial class Person

{

public void showDetails() {

Console.WriteLine("FirstName :" +firstName+ "LastName :" + lastName);

Console.ReadKey();

}

}

}

using System;

using System.Collections.Generic;

using System.Diagnostics.CodeAnalysis;

using System.Linq;

using System.Runtime.CompilerServices;

using System.Text;

using System.Threading.Tasks;

namespace Assignment2

{

internal class Program

{

static void Main(string[] args)

{

Person p1 = new Person("Ritik", "Kaushik");

p1.showDetails();

}

}

}

**6 - Create a partial class Employee with properties representing employee details. In another part, implement methods for calculating salary based on different factors.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment2

{

public partial class Employee

{

public int id;

public string name;

public double salary;

public Employee(int id, string name, double salary)

{

this.id = id;

this.name = name;

this.salary = salary;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment2

{

public partial class Employee

{

public void displayData()

{

Console.WriteLine("id :" + this.id);

Console.WriteLine("Name :" + this.name);

Console.WriteLine("Salary :" + this.salary);

Console.ReadKey();

}

}

}

using System;

using System.Collections.Generic;

using System.Diagnostics.CodeAnalysis;

using System.Linq;

using System.Runtime.CompilerServices;

using System.Text;

using System.Threading.Tasks;

namespace Assignment2

{

internal class Program

{

static void Main(string[] args)

{

Employee e1 = new Employee(1, "Ritik", 1000);

e1.displayData();

}

}

}

**7- Define an abstract base class Shape with an abstract method CalculateArea. Derive classes like Circle and Rectangle from Shape and implement the area calculation methods for each**.

using System;

using System.Collections.Generic;

using System.Diagnostics.CodeAnalysis;

using System.Linq;

using System.Runtime.CompilerServices;

using System.Text;

using System.Threading.Tasks;

namespace Assignment2

{

internal class Program

{

static void Main(string[] args)

{

Circle c1 = new Circle();

c1.setRadius(5);

c1.getarea();

}

}

public abstract class Shape

{

public abstract void getarea();

}

public class Circle : Shape

{

float radius;

public void setRadius( float r)

{

radius = r;

}

public override void getarea()

{

double area = 3.14 \* radius \* radius;

Console.WriteLine("Area is :" + area);

Console.ReadKey();

}

}

}

**8 - Design an abstract class Animal with properties like Name and Age. Derive classes like Dog and Cat from Animal with their unique methods.**

using System;

using System.Collections.Generic;

using System.Diagnostics.CodeAnalysis;

using System.Linq;

using System.Runtime.CompilerServices;

using System.Text;

using System.Threading.Tasks;

namespace Assignment2

{

internal class Program

{

static void Main(string[] args)

{

Dog d1 = new Dog();

Cat c1 = new Cat();

d1.Sound("WOW WOW");

c1.Sound("MEOW MEOW");

}

}

public abstract class Animal

{

public abstract void Sound(string sound);

}

public class Dog : Animal

{

public override void Sound(string sound)

{

Console.WriteLine("Dog is barking " + sound);

Console.ReadKey();

}

}

public class Cat: Animal

{

public override void Sound(string sound)

{

Console.WriteLine("Cat Sound " + sound);

Console.ReadKey();

}

}

}

**9- Create a base class Vehicle with methods like StartEngine and StopEngine. Derive a class Car from Vehicle and seal it. Try to create a class that inherits from Car and observe the behavior.**

using System;

using System.Collections.Generic;

using System.Diagnostics.CodeAnalysis;

using System.Linq;

using System.Runtime.CompilerServices;

using System.Text;

using System.Threading.Tasks;

namespace Assignment2

{

internal class Program

{

static void Main(string[] args)

{

Car Audi = new Car("Audi");

}

}

public sealed class Vechile

{

string vechile;

public void startEngine(string vechile)

{

this.vechile = vechile;

Console.WriteLine(vechile + " is start");

}

public void stopEngine()

{

Console.WriteLine(vechile + "is stop");

}

}

public class Car : Vechile

{

string name;

public Car(string name)

{

this.name = name;

}

}

}

**10- Design a class BankAccount with properties like AccountNumber and Balance. Implement a sealed class SavingsAccount that extends BankAccount with methods for interest calculation.**

using System;

using System.Collections.Generic;

using System.Diagnostics.CodeAnalysis;

using System.Linq;

using System.Runtime.CompilerServices;

using System.Text;

using System.Threading.Tasks;

namespace Assignment2

{

internal class Program

{

static void Main(string[] args)

{

}

sealed class Bank

{

public int AccountNumber;

public string name;

public double balance;

public Bank(int AccountNumber, string name, double balance)

{

this.AccountNumber = AccountNumber;

this.name = name;

this.balance = balance;

}

public void deposit(double balance)

{

balance += balance;

}

public void withdrawl(double amount)

{

if (amount > balance)

{

Console.WriteLine("Insufficinent Balance");

}

else

{

balance = balance - amount;

}

}

public void displayData()

{

Console.WriteLine("AccountNumber:" + this.AccountNumber);

Console.WriteLine("Name :" + this.name);

Console.WriteLine("Balance :" + this.balance);

Console.ReadKey();

}

}

class SavingAccount : Bank

{

public int AccountNumber;

public string name;

}

}

}

**Assignment 3:**

1. **Create a class with a private field and a public method to set its value.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment3

{

internal class Program

{

static void Main(string[] args)

{

User u1= new User();

u1.setName("Ritik Kaushik");

Console.WriteLine(u1.getName());

Console.ReadLine();

}

}

class User

{

private String name;

public void setName(String name)

{

this.name = name;

}

public String getName()

{

return this.name;

}

}

}

1. **Design a class with a public method that accesses a private field from another class in the same assembly**.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment3

{

internal class Program

{

static void Main(string[] args)

{

Login l1 = new Login();

l1.setNameLogin("Ritik");

Console.WriteLine(l1.getName());

Console.ReadLine();

}

}

class User

{

private String name;

public void setName(String name)

{

this.name = name;

}

public String getName()

{

return this.name;

}

}

class Login : User

{

public void setNameLogin(String name)

{

this.setName(name);

}

}

}

1. **Implement a class with an internal field and access it from a different assembly.**

using Assignment3;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace As3

{

internal class Program1

{

static void Main(string[] args)

{

Assignment3.Program program = new Assignment3.Program();

Console.WriteLine(program.name);

}

}

}

1. **Create a base class with protected members and derive a class to access those members.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment3

{

internal class Program

{

static void Main(string[] args)

{

Login l1 = new Login();

l1.setNameLogin("Ritik");

Console.WriteLine(l1.getName());

Console.ReadLine();

}

}

class User

{

protected String name;

public void setName(String name)

{

this.name = name;

}

public String getName()

{

return this.name;

}

}

class Login : User

{

public void setNameLogin(String name)

{

this.name = name;

}

}

}

1. **Develop a class with a protected internal member and create an instance in a different assembly to access it.**

Assembly1-

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.CompilerServices;

using System.Text;

using System.Threading.Tasks;

[assembly:InternalsVisibleTo("As3")]

namespace Assignment3

{

internal class Program

{

internal String name = "HRK";

protected internal void getName()

{

Console.WriteLine(name);

}

}

}

Assembly2-

using Assignment3;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace As3

{

internal class Program1

{

static void Main(string[] args)

{

Assignment3.Program program = new Assignment3.Program();

program.getName();

Console.ReadLine();

}

}

}

1. **Create a class with a public property. Derive another class and try to access the property.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment3

{

internal class Program

{

static void Main(string[] args)

{

Login l1 = new Login();

l1.setNameLogin("Ritik");

Console.WriteLine(l1.getName());

Console.ReadLine();

}

}

class User

{

public String name;

public void setName(String name)

{

this.name = name;

}

public String getName()

{

return this.name;

}

}

class Login : User

{

public void setNameLogin(String name)

{

this.name = name;

}

}

}

1. **Build a class with a private property and provide a public method to modify its value. Test the functionality.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment3

{

internal class Program

{

static void Main(string[] args)

{

User l1 = new User();

l1.setName("Ritik");

Console.WriteLine(l1.getName());

Console.ReadLine();

}

}

class User

{

public String name="HRK";

public void setName(String name)

{

this.name = name;

}

public String getName()

{

return this.name;

}

}

}

1. **Develop a class with a private method. Provide a public method that calls the private method.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment3

{

internal class Program

{

static void Main(string[] args)

{

User l1 = new User();

l1.setProfile("Ritik",20);

l1.getProfile();

Console.ReadLine();

}

}

class User

{

public String name="HRK";

private int age;

public void setProfile(String name,int age)

{

this.name = name;

this.setAge(age);

}

private void setAge(int age)

{

this.age = age;

}

public void getProfile()

{

Console.WriteLine("NAME :" + name + " " + "Age :" + age);

}

}

}

1. **Implement a class with a protected method. Derive a class and access the protected method.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment3

{

internal class Program

{

static void Main(string[] args)

{

Login l1 = new Login();

l1.setNameLoginDetails("Ritik", 21);

l1.getProfile();

Console.ReadLine();

}

}

class User

{

public String name="HRK";

private int age;

protected void setProfile(String name,int age)

{

this.name = name;

this.setAge(age);

}

private void setAge(int age)

{

this.age = age;

}

public void getProfile()

{

Console.WriteLine("NAME :" + name + " " + "Age :" + age);

}

}

class Login : User

{

public void setNameLoginDetails(String name,int age)

{

this.setProfile(name,age);

}

}

}

**10- A class with an internal constructor and create an instance from another assembly.**

Assembly1-

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.CompilerServices;

using System.Text;

using System.Threading.Tasks;

[assembly:InternalsVisibleTo("As3")]

namespace Assignment3

{

internal class Program

{

internal String name;

internal Program(String name)

{

this.name= name;

}

protected internal void getName()

{

Console.WriteLine(name);

}

}

}

Assembly2-

using Assignment3;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace As3

{

internal class Program1

{

static void Main(string[] args)

{

Assignment3.Program program = new Assignment3.Program("Ritik");

program.getName();

Console.ReadLine();

}

}

}

**Assignment 5 :**

**1. Create a C# program that intentionally throws a DivideByZeroException when dividing by zero. Catch the exception and handle it gracefully.**

using System;

public class DivideByZeroExample

{

public static void Main()

{

// Intentionally throw a DivideByZeroException.

int numerator = 10;

int denominator = 0;

try

{

int quotient = numerator / denominator;

}

catch (DivideByZeroException ex)

{

// Handle the exception gracefully.

Console.WriteLine("Cannot divide by zero.");

Console.WriteLine(ex.Message);

}

}

}

**2. Write a program that attempts to access an array element at an index that is out of bounds. Use a try-catch block to handle the IndexOutOfRangeException.**

using System;

public class IndexOutOfRangeExample

{

public static void Main()

{

// Create an array with 5 elements.

int[] myArray = new int[5];

// Attempt to access an array element at an index that is out of bounds. try

{

int element = myArray[5];

}

catch (IndexOutOfRangeException ex)

{

// Handle the IndexOutOfRangeException gracefully.

Console.WriteLine("Index out of bounds.");

Console.WriteLine(ex.Message);

}

}

}

**3. Create a C# program that uses a try-catch block to handle an exception when converting a string to an integer using int.Parse(). Handle the FormatException that may occur.**

using System;

public class ConvertStringToIntExample

{

public static void Main()

{

// Get a string from the user.

string userInput = Console.ReadLine();

// Try to convert the string to an integer. try

{

int integerValue = int.Parse(userInput);

}

catch (FormatException ex)

{

// Handle the FormatException gracefully.

Console.WriteLine("Invalid input. Please enter a valid integer.");

Console.WriteLine(ex.Message);

}

}

}

**4. Implement a C# program that uses a custom exception class. Create a custom exception and throw it in your code when a specific condition is met.**

using System;

public class CustomExceptionExample

{

public class InvalidAgeException : Exception

{

public InvalidAgeException()

{

}

public InvalidAgeException(string message)

: base(message)

{

}

}

public static void Main()

{

// Get the user's age. int age = int.Parse(Console.ReadLine());

// Throw a custom exception if the user's age is less than 18.

if (age < 18)

{

throw new InvalidAgeException("You must be at least 18 years old to use this application.");

}

Console.WriteLine("Welcome to the application!");

}

}

**5. Build a C# program that demonstrates the use of multiple catch blocks for different exception types. Handle exceptions such as IndexOutOfRangeException,**

**FormatException, and InvalidOperationException**

using System;

public class MultipleCatchBlocksExample

{

public static void Main()

{

// Try to access an array element at an index that is out of bounds. try

{

int[] myArray = new int[5];

int element = myArray[5];

}

catch (IndexOutOfRangeException ex)

{

// Handle the IndexOutOfRangeException.

Console.WriteLine("Index out of bounds.");

Console.WriteLine(ex.Message);

}

// Try to convert a string to an integer using int.Parse(). try

{

string userInput = Console.ReadLine();

int integerValue = int.Parse(userInput);

}

catch (FormatException ex)

{

// Handle the FormatException.

Console.WriteLine("Invalid input. Please enter a valid integer.");

Console.WriteLine(ex.Message);

}

// Try to perform an operation that is not valid on the current object state. try

{

object myObject = null;

myObject.ToString();

}

catch (InvalidOperationException ex)

{

// Handle the InvalidOperationException.

Console.WriteLine("Invalid operation.");

Console.WriteLine(ex.Message);

}

// Catch any other exceptions that may occur. catch (Exception ex)

{

// Handle any other exceptions.

Console.WriteLine("An unexpected error occurred.");

Console.WriteLine(ex.Message);

}

}

}

**6. Create a C# program that includes nested try-catch blocks. Throw an exception in an inner block and catch it in the outer block. Explain the flow of execution.**

using System;

public class NestedTryCatchExample

{

public static void Main()

{

// Try to perform an operation that may throw an exception. try {

// Try to perform another operation that may throw an exception. try

{

// Throw an exception.

throw new Exception("An exception occurred in the inner try block.");

}

catch (Exception ex)

{

// Handle the exception in the inner try block.

Console.WriteLine("Handled the exception in the inner try block.");

Console.WriteLine(ex.Message);

}

}

catch (Exception ex)

{

// Handle the exception in the outer try block.

Console.WriteLine("Handled the exception in the outer try block.");

Console.WriteLine(ex.Message);

}

}

}

**7. Implement a program that divides two numbers entered by the user. Handle exceptions like division by zero and invalid input. Continue to prompt the user for valid input until a valid division is performed.**

using System;

public class DivideNumbersExample

{

public static void Main()

{

// Declare variables. int numerator; int denominator;

int quotient;

// Prompt the user for the numerator. Console.Write("Enter the numerator: ");

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

// Prompt the user for the denominator. Console.Write("Enter the denominator: "); denominator = int.Parse(Console.ReadLine());

// Continue to prompt the user for valid input until a valid division is performed.

while (true)

{ try {

// Divide the numerator by the denominator.

quotient = numerator / denominator;

// Break out of the loop if the division was successful. break;

}

catch (DivideByZeroException ex)

{

// Handle the divide by zero exception.

Console.WriteLine("Division by zero is not allowed.");

Console.WriteLine(ex.Message);

// Prompt the user for the denominator again. Console.Write("Enter a valid denominator: "); denominator = int.Parse(Console.ReadLine());

}

catch (FormatException ex)

{

// Handle the invalid input exception.

Console.WriteLine("Invalid input. Please enter valid integers.");

Console.WriteLine(ex.Message);

// Prompt the user for the numerator and denominator again. Console.Write("Enter the numerator: ");

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

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

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

}

}

// Display the result of the division.

Console.WriteLine("The quotient is: {0}", quotient);

}

}

**8. Develop a C# program that demonstrates how to use the throw statement to rethrow an exception. Catch the rethrown exception and handle it appropriately**

using System;

public class RethrowExceptionExample

{

public static void Main()

{

// Try to divide two numbers. try

{

DivideByZero(10, 0);

}

catch (DivideByZeroException ex)

{

// Rethrow the exception.

throw;

}

catch (Exception ex)

{

// Handle any other exceptions.

Console.WriteLine("An unexpected error occurred.");

Console.WriteLine(ex.Message);

}

}

private static void DivideByZero(int numerator, int denominator)

{

// Try to divide the numerator by the denominator. try

{

int quotient = numerator / denominator;

}

catch (DivideByZeroException ex)

{

// Throw the exception.

throw;

}

}

}

**9. Develop a program that simulates a simple calculator with basic arithmetic operations (addition, subtraction, multiplication, and division). Use exception handling to catch and handle various type of exceptions that may occur.**

using System;

public class Calculator

{

public static double Add(double num1, double num2)

{

return num1 + num2;

}

public static double Subtract(double num1, double num2)

{

return num1 - num2;

}

public static double Multiply(double num1, double num2)

{

return num1 \* num2;

}

public static double Divide(double num1, double num2)

{

// Check for division by zero. if (num2 == 0)

{

throw new DivideByZeroException("Division by zero is not allowed.");

}

return num1 / num2;

}

public static void Main()

{

// Get the two numbers from the user. Console.Write("Enter the first number: ");

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

Console.Write("Enter the second number: ");

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

// Perform the operation based on the user's selection.

Console.WriteLine("Select an operation:");

Console.WriteLine("1. Add");

Console.WriteLine("2. Subtract");

Console.WriteLine("3. Multiply");

Console.WriteLine("4. Divide");

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

double result; try

{

switch (selection)

{ case 1:

result = Add(num1, num2); break; case 2:

result = Subtract(num1, num2);

break; case 3:

result = Multiply(num1, num2);

break; case 4:

result = Divide(num1, num2); break; default:

throw new Exception("Invalid selection.");

}

}

catch (DivideByZeroException ex)

{

Console.WriteLine(ex.Message);

return;

}

catch (FormatException ex)

{

Console.WriteLine(ex.Message);

return;

}

catch (Exception ex)

{

Console.WriteLine("An unexpected error occurred."); Console.WriteLine(ex.Message);

return;

}

// Display the result.

Console.WriteLine("The result is: {0}", result);

}

}

**Question-10. You are developing a simple e-commerce application in C#. One of the features is a shopping cart that allows users to add items to their cart. The cart is represented as an array of integers, where each integer corresponds to an item's price. Users can input the price of an item they want to add to the cart. You want to handle exceptions gracefully to ensure a smooth user experience. If the user enters an invalid price, your code should catch and handle the exception appropriately. Question: Write a C# program that simulates adding items to a shopping cart. The program should take user input for the price of items and store them in an array. Implement exception handling with multiple catch blocks to handle various scenarios. Specifically, you should handle the following exceptions: • If the user enters a negative price, catch and handle the exception as a "NegativePriceException." Display a message indicating that the price entered is invalid. • If the user enters a non-numeric value (e.g., a string), catch and handle the exception as a "FormatException." Display a message indicating that the input is not a valid price. • If the user enters a price that exceeds a predefined maximum value (e.g., 10000), catch and handle the exception as a "PriceTooHighException." Display a message indicating that the price entered is too high. Note: The program should continue to prompt the user for prices until a valid price is entered. After each valid price is entered, add it to the shopping cart array. Once the user is done adding items, display the total price of the items in the cart. Ensure that the program uses multiple catch blocks to handle the specific exceptions mentioned above and provides informative error messages.**

using System;

public class ShoppingCart

{

private const int MAX\_PRICE = 10000;

public static void Main()

{

// Create an array to store the prices of the items in the cart.

int[] cart = new int[0];

// Prompt the user to enter the price of an item. Console.Write("Enter the price of the item: "); int price = 0;

// Continue to prompt the user for prices until a valid price is entered.

while (true)

{ try

{

// Parse the user input as an integer. price = int.Parse(Console.ReadLine());

// Validate the price.

if (price < 0)

{

// Throw a NegativePriceException exception.

throw new NegativePriceException("The price entered is invalid. It cannot be negative.");

}

else if (!int.TryParse(Console.ReadLine(), out price))

{

// Throw a FormatException exception.

throw new FormatException("The input is not a valid price. It must be a number."); }

else if (price > MAX\_PRICE)

{

// Throw a PriceTooHighException exception. throw new PriceTooHighException("The price entered is too high. It cannot exceed 10000.");

}

// Add the price to the shopping cart array. Array.Resize(ref cart, cart.Length + 1); cart[cart.Length - 1] = price;

// Ask the user if they want to add another item to the cart.

Console.WriteLine("Do you want to add another item to the cart? (Y/N)"); string answer = Console.ReadLine();

if (answer != "Y")

{

break;

}

}

catch (NegativePriceException ex)

{

// Display an error message to the user.

Console.WriteLine(ex.Message);

}

catch (FormatException ex)

{

// Display an error message to the user.

Console.WriteLine(ex.Message);

}

catch (PriceTooHighException ex)

{

// Display an error message to the user.

Console.WriteLine(ex.Message);

}

catch (Exception ex)

{

// Handle any other unexpected exceptions.

Console.WriteLine("An unexpected error occurred."); Console.WriteLine(ex.Message);

return;

}

}

// Calculate the total price of the items in the cart.

int totalPrice = 0;

foreach (int itemPrice in cart)

{

totalPrice += itemPrice;

}

// Display the total price to the user.

Console.WriteLine("The total price of the items in your cart is: {0}", totalPrice);

}

}

public class NegativePriceException : Exception

{

public NegativePriceException(string message) : base(message)

{

}

}

public class FormatException : Exception

{

public FormatException(string message) : base(message)

{

}

}

public class PriceTooHighException : Exception

{

public PriceTooHighException(string message) : base(message)

{

}

}