

M3.....	1
Exception handling-:.....	1
Out of memory Exception-:	2
Code-:.....	2
Arithmetic Exception-:.....	3
Code-:.....	3
Divide By Zero Exception-:.....	4
Code-:.....	4
Argument exception-:	5
Code-:.....	5
D2	7
Arrays-:	7
Code-:.....	7
Properties-:	8
Code-:.....	9
Structures-:.....	10
Code-:.....	10
Enum-:.....	11
Code-:.....	11
Delegates-:	12
Code-:.....	13

M3

In a Written report, present a sample code in C# for error handling concept. You must include a minimum of four system imposed exceptions in your coding with the screenshots of your results.

Exception handling-:

Exception is problem that arises during the execution of program. And exception is an response to an exceptional, which will arise at the time of running of program like for example when I am diving the number with zero and it provides the solution for the program in the form of try, catch blocks, using these kind of exception error can be handled.

The exception handling for the systems exceptions are-:

1. Try-:

It is used to identify a block of code for which particular exceptions is activated and it is followed by more catch blocks

2. Catch-:

It will help to catch the error for clear the error in program where handle of problem is required. It will inform catching of an exception.

3. Finally-:

Which is used to execute given set of statements, while exception is given by the program or not

4. Throw-:

Which will shows an exception when the problem is shown in the program and the exception is thrown by this key word.

Out of memory Exception-:

This type of exceptions will indicates failure in the catastrophic, if the error need to handle then it should use catch block the error which got occurred and it will terminates the application which will add events logs to application

Syntax for this is-:

```
[SerializableAttribute]
[ComVisibleAttribute(true)]
public class OutOfMemoryException : SystemException
```

Code-:

```
using System;
```

```
public class Example
{
    public static void Main()
    {
        try {
            // Outer block to handle any unexpected exceptions.
            try {
                string s = "This";
                s = s.Insert(2, "is ");

                // Throw an OutOfMemoryException exception.
                throw new OutOfMemoryException();
            }
            catch (ArgumentException) {
                Console.WriteLine("ArgumentException in String.Insert");
            }

            // Execute program logic.
        }
        catch (OutOfMemoryException e) {
            Console.WriteLine("Terminating application unexpectedly...");
            Environment.FailFast(String.Format("Out of Memory: {0}",
                e.Message));
        }
    }
}
```

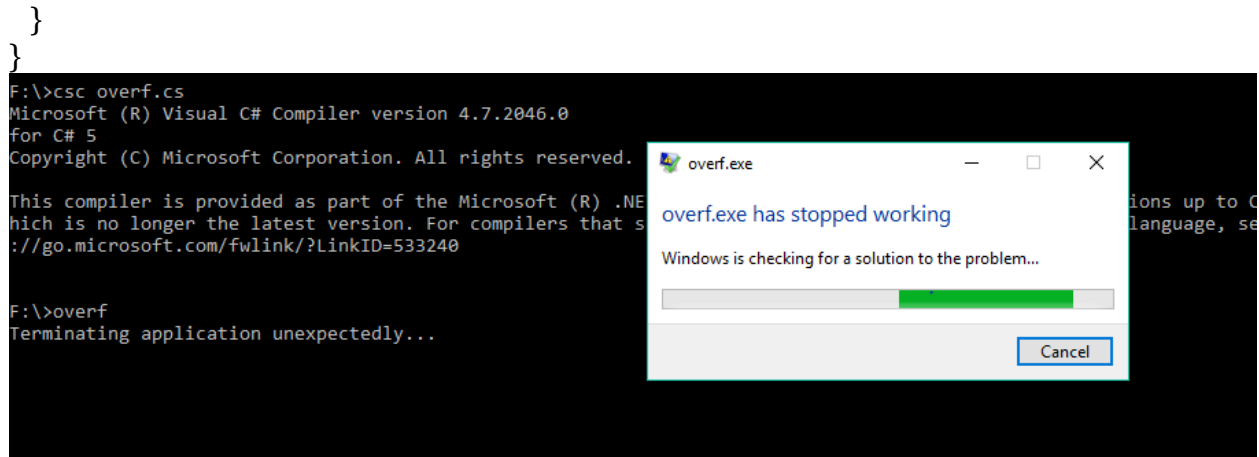


Figure 1output

Arithmetic Exception:-

This is based on divide by zero exception and other two exceptions, which means use of one derivative class of this exception and the code should not handle or throw the error but it should derive the class since it indicates the exact nature of error.

Syntax for this is:-

```
[SerializableAttribute]
[ComVisibleAttribute(true)]
public class ArithmeticException : SystemException
```

Code:-

```
using System;           //giving the namespace as system
class pushpak {         //giving class name as pushpak
    public static void Main() { //giving method
        double myNan = Double.NaN; //giving double nan
        try {
            Math.Sign(myNan);
        }
        catch (ArithmeticException e) { //exceptions
            Console.WriteLine("Error: {0}",e);
        }
    }
}
```

```
F:\>csc ari.cs
Microsoft (R) Visual C# Compiler version 4.7.2046.0
for C# 5
Copyright (C) Microsoft Corporation. All rights reserved.

This compiler is provided as part of the Microsoft (R) .NET Framework, but only supports language versions
which is no longer the latest version. For compilers that support newer versions of the C# programming langu
: //go.microsoft.com/fwlink/?LinkID=533240

F:\>ari
Error: System.ArithmeticException: Function does not accept floating point Not-a-Number values.
   at System.Math.Sign(Double value)
   at pushpak.Main()
```

Figure 2output

Divide By Zero Exception:-

It will try to divide integer or decimal number with the zero, then these exception is seen for avoid these kind of error the denominator should not be zero and diving the values with float values doesn't throw the error.

Syntax of this exception:-

```
[SerializableAttribute]
[ComVisibleAttribute(true)]
public class DivideByZeroException : ArithmeticException
```

Code:-

```
using System;    //giving the namespace as system
```

```
namespace Error {
```

```
    class Div {    //class name giving as Div
        int res;
```

```
        Div() {
            res = 0;
        }
```

```
        public void division(int number1, int number2) {
            try {
                res = number1 / number2;    //diving code
            } catch (DivideByZeroException e) {
                Console.WriteLine("Exception caught: {0}", e);
            } finally {
                Console.WriteLine("Result: {0}", res);
            }
        }
    }
```

```
    static void Main(string[] args) {    //method given
        Div d = new Div();
```

```
d.division(25, 0);
Console.ReadKey();
}
}
}
```

```
F:\>csc arra.cs
Microsoft (R) Visual C# Compiler version 4.7.2046.0
for C# 5
Copyright (C) Microsoft Corporation. All rights reserved.

This compiler is provided as part of the Microsoft (R) .NET Framework, but only supports la
which is no longer the latest version. For compilers that support newer versions of the C# p
://go.microsoft.com/fwlink/?LinkID=533240

F:\>arra
Exception caught: System.DivideByZeroException: Attempted to divide by zero.
   at Error.Div.division(Int32 number1, Int32 number2)
Result: 0
```

Figure 3output

Argument exception:-

When exception is thrown when null or empty reference is passing to a method that does not accept valid argument and if the method can be called with invalid arguments and this is thrown in the null, empty case and these does not give them extra abilities.

Syntax for this is:-

[SerializableAttribute]

[ComVisibleAttribute(true)]

public class ArgumentNullException : ArgumentException

Code:-

using System;

```
class Program
{
    static void Main()
    {
        // Demonstrate the argument null exception.
        try
        {
            A(null);
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex);
        }
        // Demonstrate the general argument exception.
        try
```

```

    {
        A("");
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex);
    }
    // Flow path without exception.
    Console.WriteLine(A("test"));
}

static int A(string argument)
{
    // Handle null argument.
    if (argument == null)
    {
        throw new ArgumentNullException("argument");
    }
    // Handle invalid argument.
    if (argument.Length == 0)
    {
        throw new ArgumentException("Zero-length string invalid", "argument");
    }
    return argument.Length;
}
}

```

```

F:\>csc out.cs
Microsoft (R) Visual C# Compiler version 4.7.2046.0
for C# 5
Copyright (C) Microsoft Corporation. All rights reserved.

This compiler is provided as part of the Microsoft (R) .NET Framework, but only supports
versions that are supported by the .NET Framework. For more information, see the link
http://go.microsoft.com/fwlink/?LinkID=533240

F:\>out
System.ArgumentNullException: Value cannot be null.
Parameter name: argument
    at Program.A(String argument)
    at Program.Main()
System.ArgumentException: Zero-length string invalid
Parameter name: argument
    at Program.A(String argument)
    at Program.Main()
4
F:\>

```

Figure 4output

D2

You must develop C# programs using multiple code elements.

Your code must include:

- Arrays (Determine any two integer in an array sum to the target integer)
- Properties (Write a simple program to implement properties)
- Structures (Write a simple program to implement structures)
- Enum (Create an enum named temperature with values Low, Medium & high. Get the temperature and return the result)
- Delegates. (Write a program for delegate and return the value)
- Present the same with output & screenshots with appropriate annotations in a written report.

Arrays:-

It stores the fixed type of collections of elements of same type and it is used to store collection of data, instead of declaring individual in the array the developer can declare one array and the specific element in the array is accessed by index and it consists of lot of memory locations, the lower and highest address is given to elements.

Example:-

```
double[] balance = new double[10];
balance[0] = 4500.0;
```

Code:-

```
using System; //giving system as namespace
class pushpak //using class as pushpak
{
    public static void Main() //method starts
    {
        int r,p; //giving r as first number and p as second number
        int[] numbers = new int[10] {0,1,2,4,6,3,5,7,8,9}; //declaring array
        number
        Console.WriteLine("Target number: ");
        string number=Console.ReadLine();
        int target_number=int.Parse(number);
        for(r=0;r<10;r++) //first number
        {
            for(p=1;p<10;p++) //second number
            {
                int sum=numbers[r]+numbers[p]; // checking the target
                number of their sum
                if(sum == target_number)
                {
                    Console.WriteLine("The sum of "+numbers[r]+" and "+numbers[p]+" is target
                    number"); //for the display of target number
                }
            }
        }
    }
}
```

```

}
This compiler is provided as part of the Microsoft (R) .NET Framework, but only supports language version 4.0 which is no longer the latest version. For compilers that support newer versions of the C# programming language, see http://go.microsoft.com/fwlink/?LinkID=533240
F:\>arr
Target number: 10
The sum of 1 and 9 is target number
The sum of 2 and 8 is target number
The sum of 4 and 6 is target number
The sum of 6 and 4 is target number
The sum of 3 and 7 is target number
The sum of 5 and 5 is target number
The sum of 7 and 3 is target number
The sum of 8 and 2 is target number
The sum of 9 and 1 is target number
F:\>

```

Figure 5 output array code

Properties:-

It is the member which provides constant mechanism to read, write or compute the private fields and it can be used as if they are public members but they actually called Assessors which enable data to be access easily to the developers and it helps safety, flexibility of methods

Example:-

```

class Example
{
    int _number;
    public int Number
    {
        get
        {
            return this._number;
        }
        set
        {
            this._number =
value;
        }
    }
}

class Program
{
    static void Main()
    {
        Example example = new
Example();
        example.Number = 5; //
set { }

Console.WriteLine(example.Number
); // get { }

```



```
    }  
}
```

Code-:

```
using System;    //giving system as namespace
namespace one {
    class Student {    //giving class as student
        private string name = "not known";
        private int age = 0;
        public string Name {    //property of string
            get {
                return name;
            }
            set {
                name = value;
            }
        }
    }
    public int Age {    //property of integer
        get {
            return age;
        }
        set {
            age = value;
        }
    }
    public override string ToString() {    //override string
        return "Name = " + Name + ", Age = " + Age;
    }
}
class Demo {
    public static void Main() {
        Student s = new Student();
        s.Name = "Zara";    //giving values
        s.Age = 9;
        Console.WriteLine("Student Info: {0}", s);    //showing output
        Console.ReadKey();
    }
}
```

```
F:\>csc prop.cs
Microsoft (R) Visual C# Compiler version 4.7.2046.0
for C# 5
Copyright (C) Microsoft Corporation. All rights reserved.

This compiler is provided as part of the Microsoft (R) .NET Framework, but only supports
versions that are no longer the latest version. For compilers that support newer versions of th
e .NET Framework, see http://go.microsoft.com/fwlink/?LinkID=533240
for more information.

F:\>prop
Student Info: Name = Zara, Age = 9
```

Figure 6 output for properties

Structures:-

It is the composite data type which consists of number of elements of other data types and it is the value type, structure that are created in the stack and it contains many structure like methods, constructors and etc. and these contains fields which can declare public, private and etc. and field inside the structure can be initialize.

Example:-

```
struct MyStruct {
    public int x;
    public int y;
}
```

Code:-

```
using System;           //giving system as namespace
struct Books {          //giving struct method
    private string title; //assigning the names
    private int book_id;
    public void getValues(string t, int id) { //code for getting values
        title = t;
        book_id = id;
    }
    public void display() { //display code
        Console.WriteLine("Title : {0}", title);
        Console.WriteLine("Book_id :{0}", book_id);
    }
};
public class testStructure { //giving class
    public static void Main(string[] args) {
        Books Book1 = new Books();
        Book1.getValues("C Programming", 6495407); //giving values
        Book1.display();
        Console.ReadKey();
    }
}
```

```
F:\>csc en.cs
Microsoft (R) Visual C# Compiler version 4.7.2046.0
for C# 5
Copyright (C) Microsoft Corporation. All rights reserved.

This compiler is provided as part of the Microsoft (R) .NET Framework, but
which is no longer the latest version. For compilers that support newer ver
://go.microsoft.com/fwlink/?LinkID=533240

F:\>en
Title : C Programming
Book_id :6495407
```

Figure 7 output for structures

Enum-:

It is used to declare an enumeration, that consists of set of named constant called enumerator list and it is defines enum directly with the namespace, so that every class can access with equal convenience and it also be test with the class, it is used to override the default values .

Example-:

```
enum Color
{
    Red,
    Blue,
}
class Shape
{
    public void Fill(Color color) {
        switch(color) {
            case Color.Red:
                ...
                break;
            case Color.Blue:
                ...
                break;
            default:
                break;
        }
    }
}
```

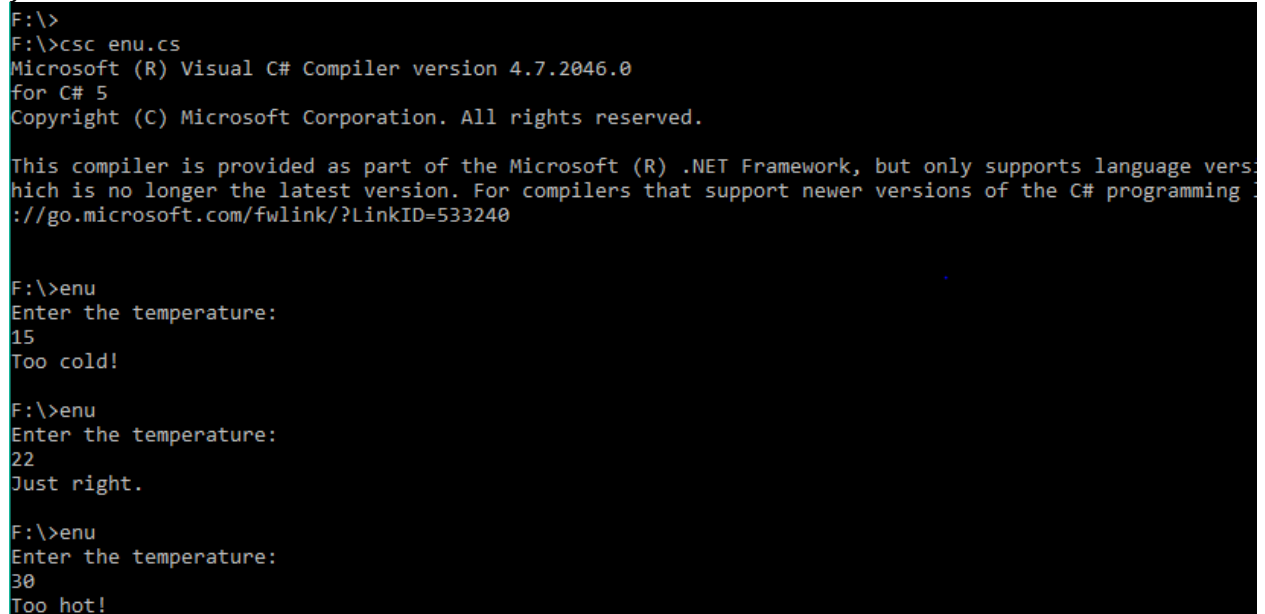
Code-:

```
using System; //giving the namespace as system
class temp //giving class name as temp
{
    enum Temperature //declaring enum
    {
        Low,
```

```

        Medium,
        High,
    };
    public static void Main() //declaring method
    {
        Console.WriteLine("Enter the temperature: ");
        string temp = Console.ReadLine();
        int temperature = int.Parse(temp);
        if(temperature<21) //using if statement
        {
            Console.WriteLine("Too cold!");
        }
        else if(temperature < 23) { //using else if statement
            Console.WriteLine("Just right.");
        }
        else { //using else statement
            Console.WriteLine("Too hot!");
        }
    }
}

```



```

F:\>
F:\>csc enu.cs
Microsoft (R) Visual C# Compiler version 4.7.2046.0
for C# 5
Copyright (C) Microsoft Corporation. All rights reserved.

This compiler is provided as part of the Microsoft (R) .NET Framework, but only supports language versions
which is no longer the latest version. For compilers that support newer versions of the C# programming language,
please see: http://go.microsoft.com/fwlink/?LinkID=533240

F:\>enu
Enter the temperature:
15
Too cold!

F:\>enu
Enter the temperature:
22
Just right.

F:\>enu
Enter the temperature:
30
Too hot!

```

Delegates:-

It is similar to the functions in C. it is reference type variable which holds reference to method and it can change according to the runtime and it is generally used for implementing events and call-back methods, it can refer to a method and that have same signature as like as of delegates and once delegate object is created with new keyword, that is associated with particular method.

Example:-

```

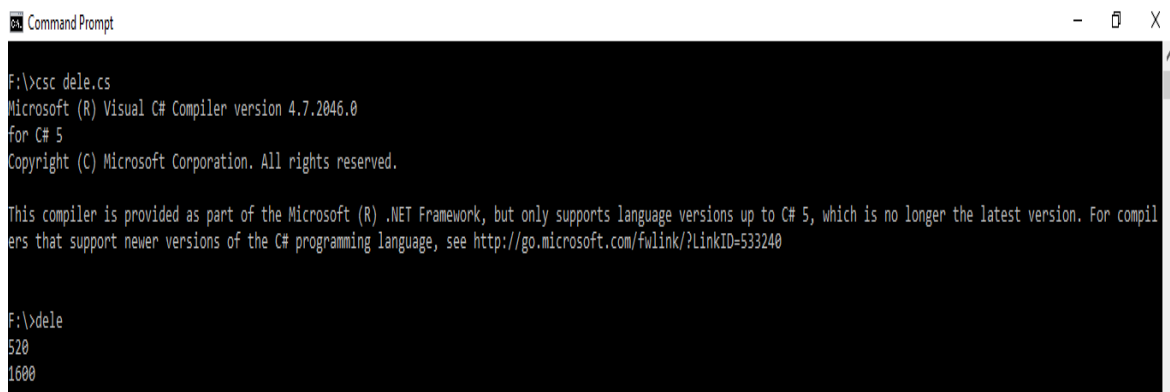
class run
{
    static void l() {

```

```
System.Console.WriteLine("Test.1");
}
static void Main() {
SimpleDelegate d = new SimpleDelegate(1);
d();
}
}
```

Code:-

```
using System;                                //giving system as namespace
delegate int MyDelegate(int num);           //delegate starts
class run                                    //giving class name as run
{
    static void Main()                       //method starts
    {
        MyDelegate del = Treble;           //declaring the first delegate method
        int result = del(65);               //giving first delegate value
        Console.WriteLine(result);
        del += Square;                     //declaring the second delegate method
        result = del(40);                   //giving second delegate value
        Console.WriteLine(result);
        Console.ReadKey();
    }
    static int Treble(int num)               //giving the first delegate method
    {
        return num * 8;
    }
    static int Square(int num)              //giving the first delegate method
    {
        return num * num;
    }
}
```



```
Command Prompt
F:\>csc dele.cs
Microsoft (R) Visual C# Compiler version 4.7.2046.0
for C# 5
Copyright (C) Microsoft Corporation. All rights reserved.

This compiler is provided as part of the Microsoft (R) .NET Framework, but only supports language versions up to C# 5, which is no longer the latest version. For compilers that support newer versions of the C# programming language, see http://go.microsoft.com/fwlink/?LinkID=533240

F:\>dele
520
1600
```

Figure 8output for delegates