1) Write a small program where you need to implement a Try and Catch Block .

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace TryCatchPrograms
{
   public class Program
   {
       public static void SomeMethod()
      {
         int[] arr = { 10, 0, 20, 30, 40, 50 };
         try
         {
            for (int i = 0; i < arr.Length; i++)
            {
               Console.WriteLine(arr[i] / arr[i + 1]);

             }
         }
         catch (IndexOutOfRangeException ex)
         {
            Console.WriteLine("An Exception has occurred : {0}", ex.Message);
         }
         catch (DivideByZeroException ex)
         {
            Console.WriteLine("An Exception has occurred : {0}", ex.Message);
         }

      }

      static void Main(string[] args)
      {
         SomeMethod();

         Console.ReadKey();
      }
   }
}
```

```
C:\Users\shalini.kumari\source\repos\TryCatchPrograms\bin\Debug\TryCatchPrograms.exe
An Exception has occurred : Attempted to divide by zero.
```

2) When should we write multiple catch blocks for a Single Try block?

1.Multiple Catch Block = Multiple catch blocks are used to handle different types of exceptions means each catch block is used to handle different types of exception.

2. Single Try Block = A try block is the block of code (Contains a set of Statements) in which exceptions can occur.

Syntax :-

```
Try
{
  // Code  that may throw an exception
}

catch(Exception)
{
  // code
}
```

3) How to define a delegate and call any method or event using it?

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DelegetPrograms1
{
    public class Program
    {
        public delegate void addnum(int a, int b);
        public delegate void subnum(int a, int b);
        public delegate void mulnum(int a, int b);
        public delegate void divnum(int a, int b);

        public void sum(int a, int b)
        {
            Console.WriteLine("(100 + 40) = {0}", a + b);
        }
        public void subtract(int a, int b)
        {
            Console.WriteLine("(100 - 60) = {0}", a - b);
        }
        public void multiply(int a, int b)
        {
            Console.WriteLine("(20 * 6) = {0}", a * b);
        }
        public void division(int a, int b)
        {
            Console.WriteLine("(100 % 20) = {0}", a % b);
        }

        static void Main(string[] args)
        {
            Program obj = new Program();

            addnum del1 = new addnum(obj.sum);
            Sub num del2 = new sub num(obj.subtract);
```

```
        Mul num del3 = new M ul num(obj.multiply);
        Div num del4 = new div num(obj.division);

        del1(100, 40);
        del2(100, 60);
        del3(20, 6);
        del4(100, 20);
        Console.ReadKey();

    }

  }
}
```

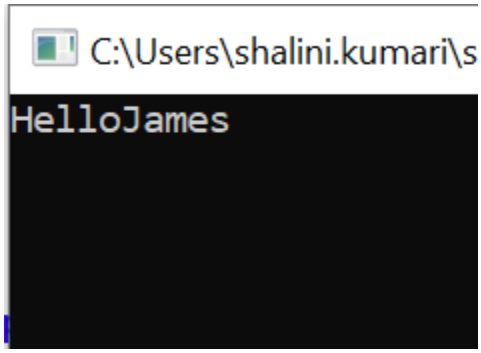
```
(100 + 40) = 140
(100 - 60) = 40
(20 * 6) = 120
(100 % 20) = 0
```

5) What will be the output of below code snipped :

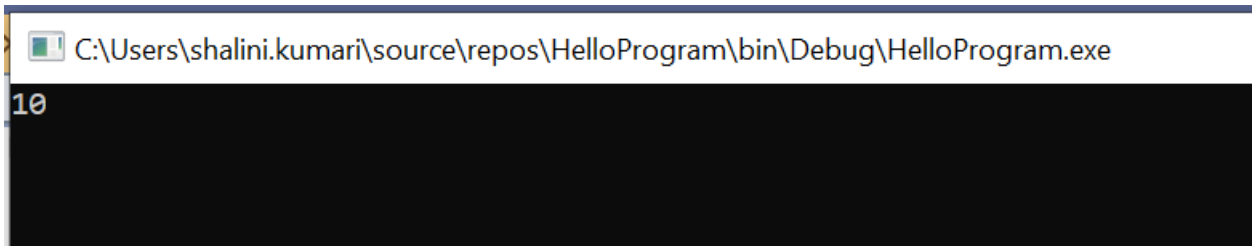
```
static void Main()
{
  Func <string,string> output=delegate(string name)
{
  return "Hello" + name;
};
Console.Write(output("James"));
}

static void Main()
{
  Action <int> output = i=>Console.Write(i);
  output(10);
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace HelloProgram
{
    public class Program
    {
        static void Main(string[] args)
        {
            Func<string, string> output = delegate (string name)
            {
                return "Hello" + name;
            };
            Console.Write(output("James"));
            Console.ReadKey();

        }
    }
```

```
}
```


C:\Users\shalini.kumari\s
HelloJames

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace HelloProgram
{
    public class Program
    {
        static void Main(string[] args)
        {
            Action<int> output = i => Console.Write(i);
            output(10);
            Console.ReadKey();

        }
    }
}
```



6) Write a program to implement Async await with proper justification.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Security.Policy;
using System.Text;
using System.Threading.Tasks;

namespace TaskBasedAsynchronousProgramming
{
    public class Employee
    {
        public int ID { get; set; }
        public string Name { get; set; }
        public int Age { get; set; }
        public double Salary { get; set; }
        public string Address { get; set; }
        public async static void SomeMethod()
        {
            Employee emp = await GetEmployeeDetails();
            Console.WriteLine($"ID: {emp.ID}, Name : {emp.Name}, Salary :
{emp.Salary}, Age :  {emp.Age}, Address : {emp.Address}");
        }
         public static async Task<Employee> GetEmployeeDetails()
        {
            Employee employee = new Employee();
            employee.ID = 1;
            employee.Name = "soni";
            employee.Age = 21;
            employee.Salary = 280000;
            employee.Address = "Delhi";
            return employee;
        }
    }
    public class program
   {
        static void Main(string[] args)
         {
           Console.WriteLine($"Main Thread Started:");
```

```
            Employee.SomeMethod();
            Console.WriteLine($"Main Thread Completed:");
            Console.ReadKey();
        }
    }
}
```

```
C:\Users\shalini.kumari\source\repos\TaskBasedAsynchronousProgramming\bin\Debug\TaskBasedAsynchronousProgramming.exe
Main Thread Started:
ID: 1, Name : soni, Salary : 280000, Age :  21, Address : Delhi
Main Thread Completed:
```

4) **Try to use Func, Action and Predicate any program.**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Func Action Predicate Program
{
    public class Program
    {
        static void Main(string[] args)
        {

            Func<int, int, int> addFunction = (a, b) => a + b;
            Console.WriteLine($"Func<int, int, int> addFunction: {addFunction(24,76)}");


            Action<string> greetAction = name => Console.WriteLine($"Hello, {name}!");
            greetAction("Ram");


            Predicate<int> isEvenPredicate = num => num % 2 == 0;
            Console.WriteLine($"Predicate<int> isEvenPredicate(2): {isEvenPredicate(8)}");
            Console.WriteLine($"Predicate<int> isEvenPredicate(3):
```

```
{isEvenPredicate(21)}");
            Console.ReadKey();
        }
    }
}
```

C:\Users\shalini.kumari\source\repos\FuncsActionPredicateProgram\bin\Debug\FuncsActionPredicateProgram.exe

```
Func<int, int, int> addFunction: 100
Hello, Ram!
Predicate<int> isEvenPredicate(2): True
Predicate<int> isEvenPredicate(3): False
```