DAY12 Assignment

By
Ram Charan

1.What is Exception Handling and why we need exception handling.

- It is done to ensure that our application should not be crashed
- It will not display any technical issues and to make sure that it handle errors gracefully and display friendly messages.

- The process of responding the occurence of exceptions is known as Exception Handling.

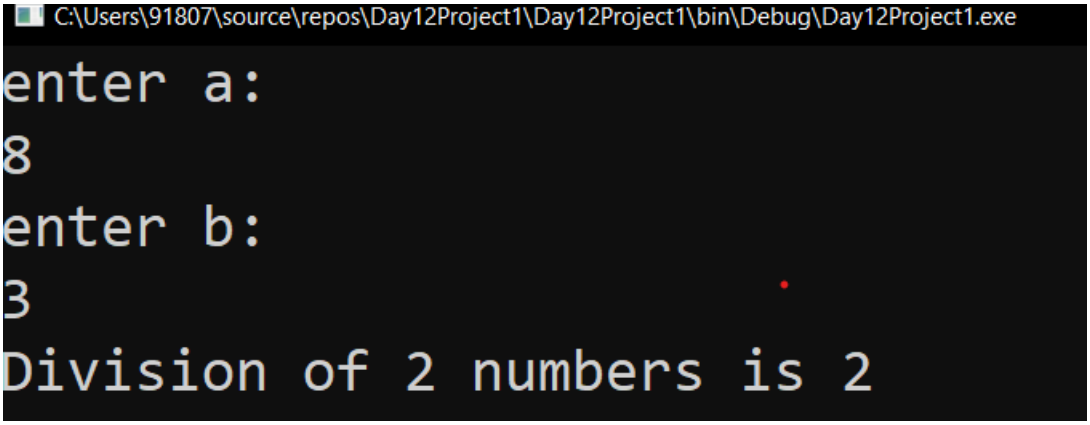
2.Write a simple division program and handle three exceptions and also add super exception at last.

Code:

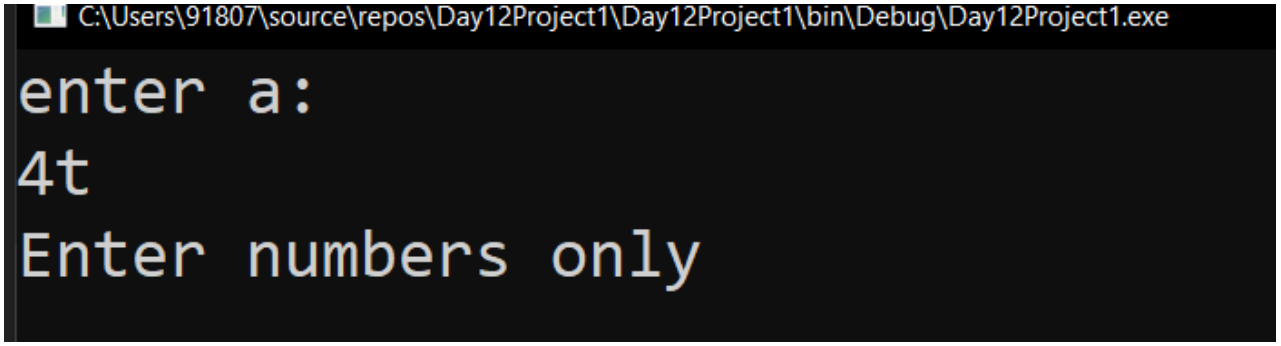
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Day12Project1
{
   //Author:Rc
   /*Purpose: Division program and handle 3 exceptions and add super exception*/
   internal class Program
   {
      static void Main(string[] args)
      {
         //Variables declaration
         int a,b, c;
         try
         {
            Console.WriteLine("enter a:");
            a = Convert.ToInt32(Console.ReadLine());
```

```
            Console.WriteLine("enter b:");
            b = Convert.ToInt32(Console.ReadLine());

            c = a / b;
            Console.WriteLine("Division of 2 numbers is {0}",c);

            Console.ReadLine();
        }
        catch (OverflowException) //It occurs when numbers are greater than range
        {
            Console.WriteLine("enter numbers between -2147483648 to 2147483647");
            Console.ReadLine();
        }
        catch(DivideByZeroException)//If number is divided by 0
        {
            Console.WriteLine("Number cant be divided by zero");
            Console.ReadLine();
        }
        catch(FormatException)//If we enter wrong input like alphabets
        {
            Console.WriteLine("Enter numbers only");
            Console.ReadLine();
        }
        catch(Exception)//It is the super
        {
            Console.WriteLine("Something went wrong, contact Rc");
            Console.ReadLine();
        }


    }
  }
}
```

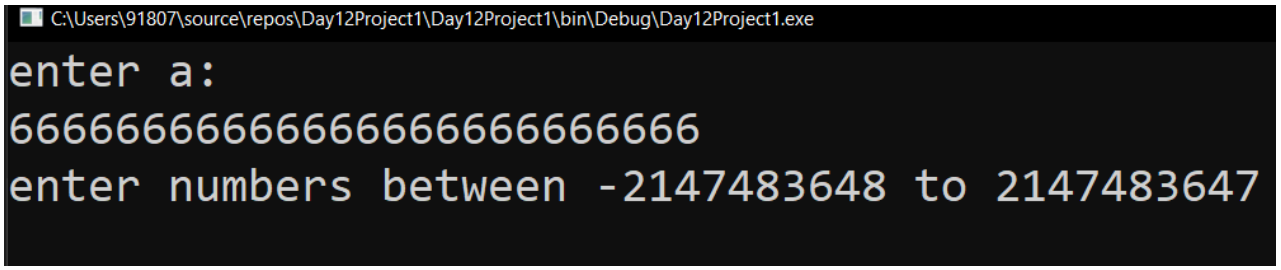
Output 1: Division Success



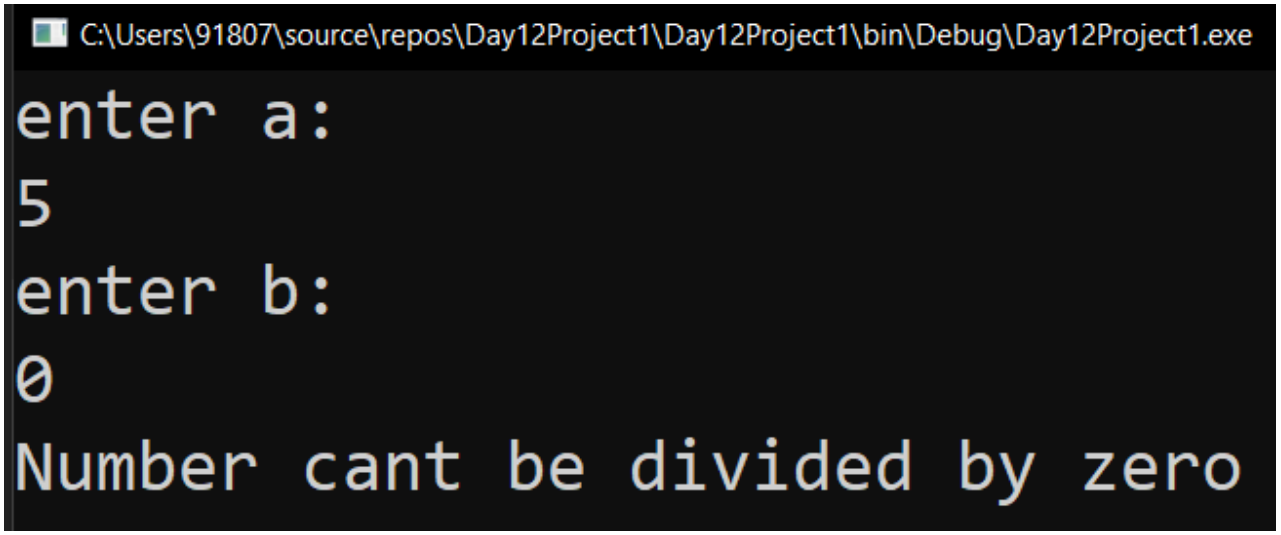
Output2: Format Exception



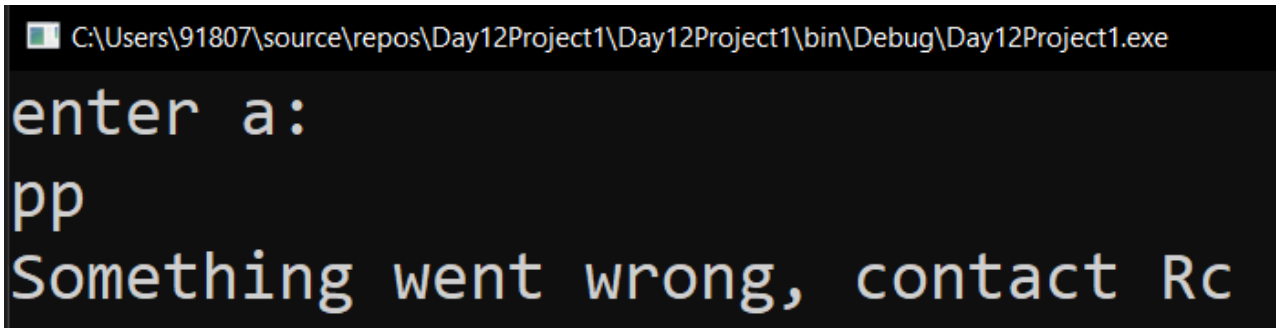
Output3:Overflow Exception



Output4:Divide by Zero Exception



Output5: Super Exception

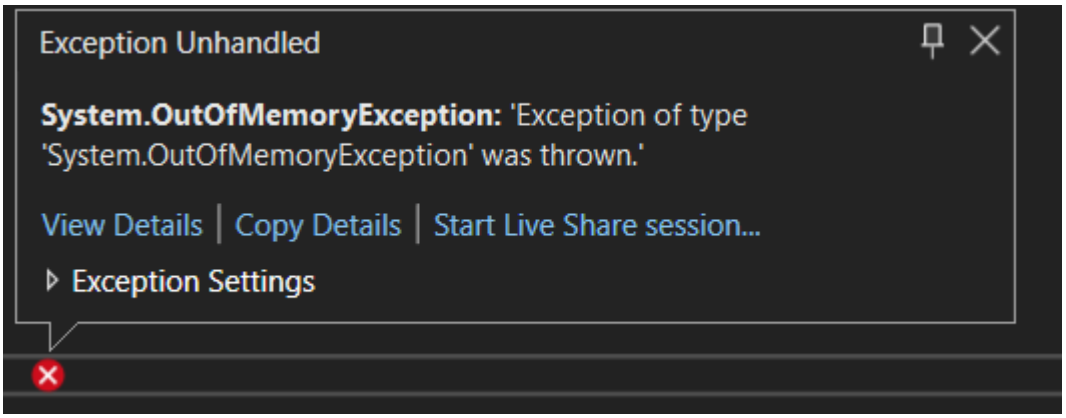


3.Research and write atleast 6 exceptions that occur in C# with sample code.

a.)OutOfMemory Exception:

Code:

```
static void Main(string[] args)
        {
            string name =new string('r',int.MaxValue);


            Console.ReadLine();
        }
```

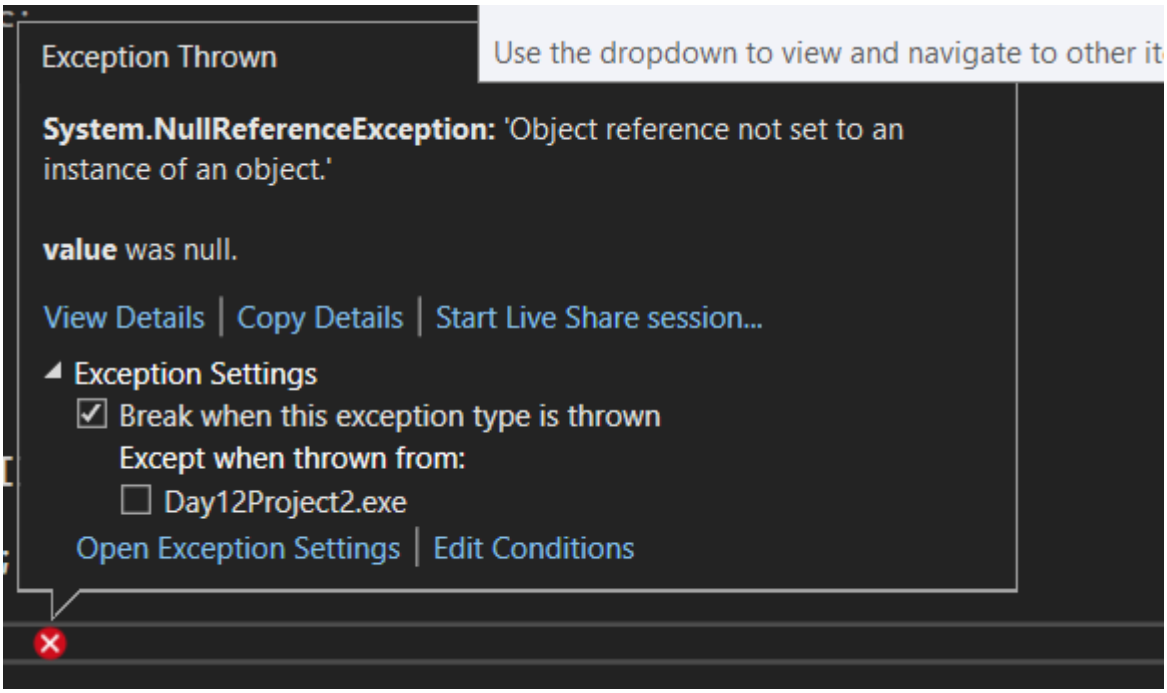


b.)NullReference Exception:

Code:

```
static void Main(string[] args)
        {
            string value = null;

            if(value.Length==0)
            {
                Console.WriteLine(value);
            }


            Console.ReadLine();
        }
```

Output:

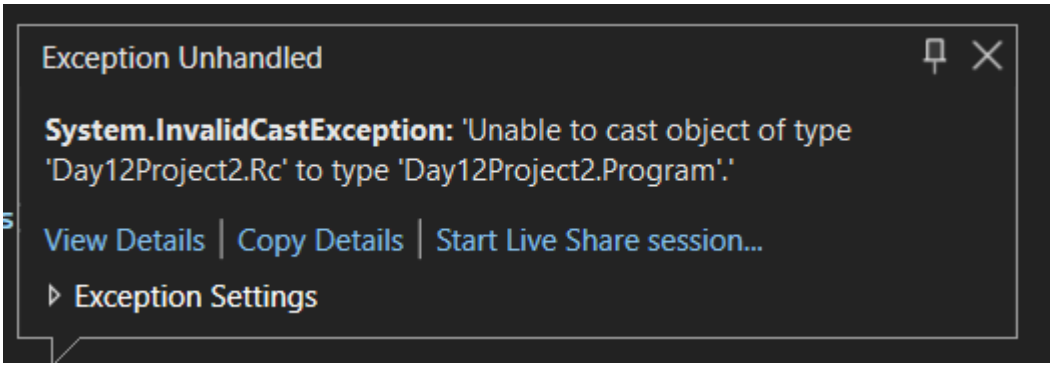


c.)InvalidCast Exception:

Code:

```
class Rc
  {

  }
  internal class Program
  {
     static void Main(string[] args)
     {
        Rc p1 = new Rc();
        object p2 = p1;
        Program p3 = (Program)p2;


        Console.ReadLine();
     }
  }
```

Output:

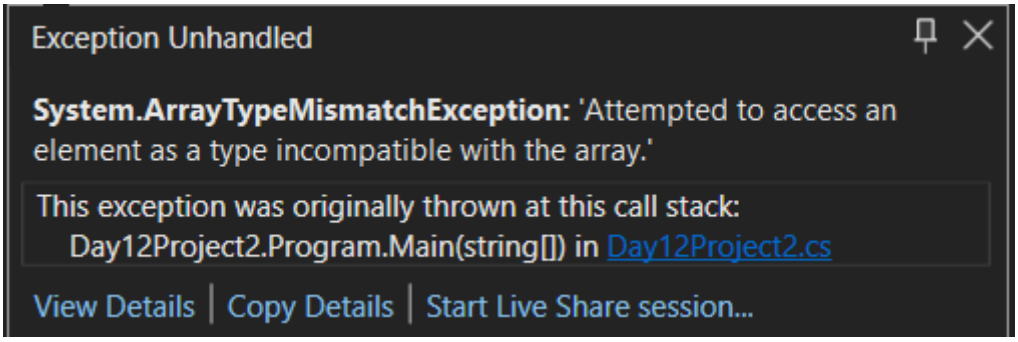


d.)ArrayTypeMismatch Exception:

Code:

```
static void Main(string[] args)
     {
        string[] a = { "Hi" };
        object[] b = a;
        b[0] = 3;


        Console.ReadLine();
     }
```

Output:

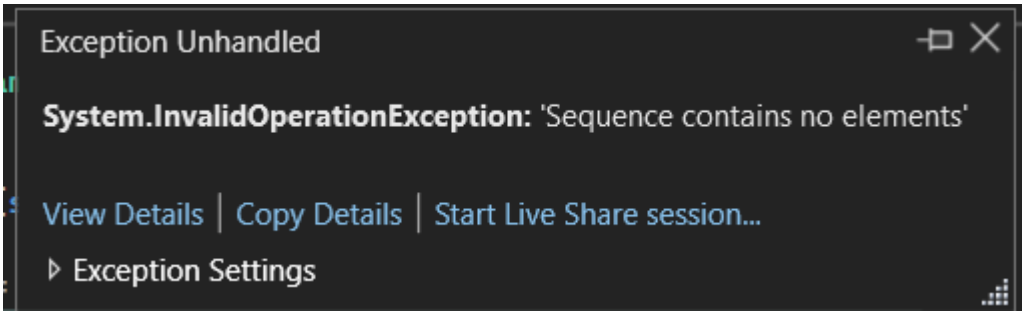


e.)InvalidOperation Exception:

Code:

```
static void Main(string[] args)
      {
         var numbers = new List<int> { 1, 3, 4 };
         var firstgreaterThanFive = numbers.Where(x => x > 5).First();

         Console.ReadLine();
      }
```

Output:



4.What is use of “finally” block illustrate with an example.

- The statements in finally block are executed irrespective of exceptions.

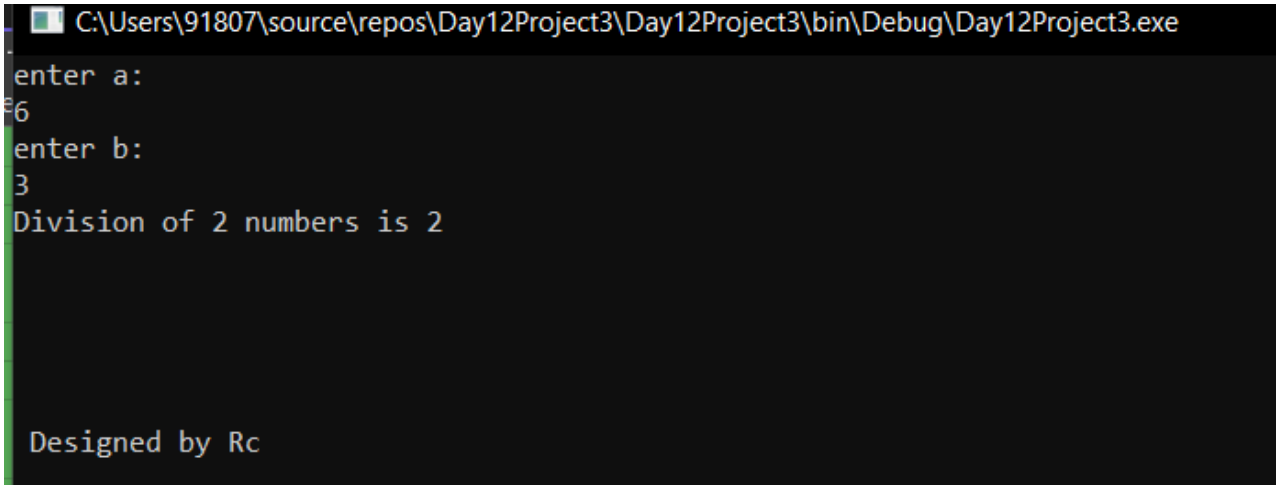
Code:

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

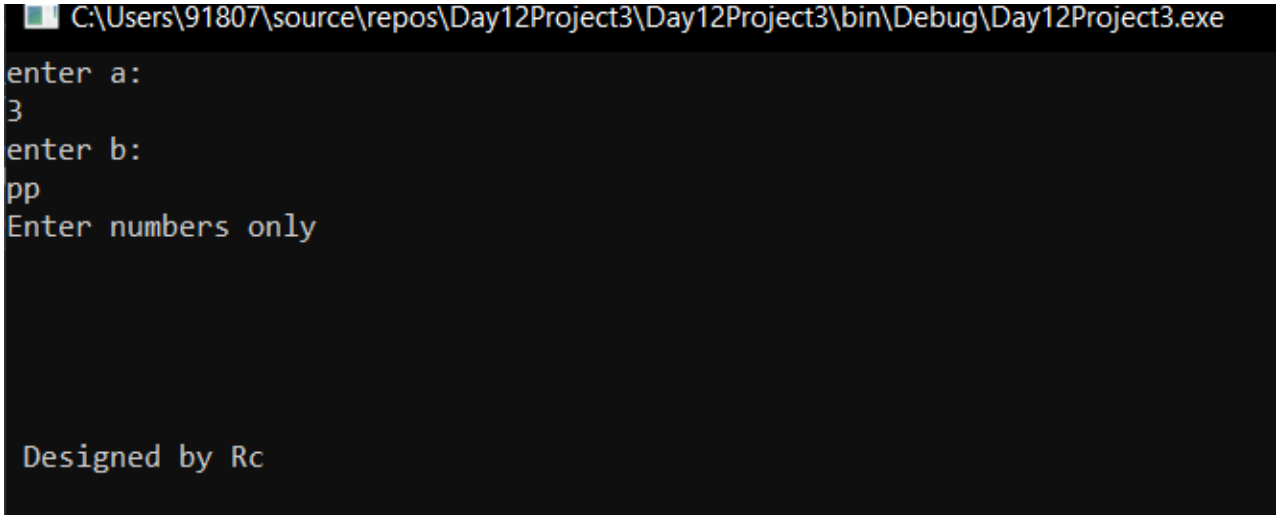
```
namespace Day12Project1
{
   //Author:Rc
   /*Purpose: Division program and handle 3 exceptions and add super exception*/
   internal class Program
   {
      static void Main(string[] args)
      {
         //Variables declaration
         int a, b, c;
         try
         {
            Console.WriteLine("enter a:");
            a = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("enter b:");
            b = Convert.ToInt32(Console.ReadLine());

            c = a / b;
            Console.WriteLine("Division of 2 numbers is {0}", c);


         }
         catch (OverflowException) //It occurs when numbers are greater than range
         {
            Console.WriteLine("enter numbers between -2147483648 to 2147483647");

         }
         catch (DivideByZeroException)//If number is divided by 0
         {
            Console.WriteLine("Number cant be divided by zero");

         }
         catch (FormatException)//If we enter wrong input like alphabets
         {
            Console.WriteLine("Enter numbers only");

         }
         catch (Exception)//It is the super exception
         {
            Console.WriteLine("Something went wrong, contact Rc");

         }
         finally
         {
            Console.WriteLine("\n \n \n \n \n Designed by Rc");
            Console.ReadLine();
         }
```

```
    }
  }
}
```

Output1:



Output2:



5.Write about exception handling.

- Exception handling is done to gracefully handle errors.
- A single try block can have multiple catch blocks.
- Always write general exception at last.
- Statements inside final block will executed irrespective of errors.
- Syntax:

```
try
  {
```

```
}
catch (Exception1)
{

}
catch(Exception2)
{

}


catch(Exception)
{

}
finally
{


}
```

6.Write differences between Compilation error and Runtime error.

Parameters	Compile time errors	Run time errors
Detection	• Compilers can easily detect compile time errors during development of code.	• Compiler doesnot detect runtime errors during the code execution.
Reference	• These errors generally refers to errors that corresponds to syntax.	• It refers to the error that occur during code execution during runtime
Fixation	• We can easily fix these errors during development of code.	• A compiler cannot identify run-time error. But we can fix it after execution of code.

7.Write 6 compilation errors with small code snippets.

1. Code: Semicolon error

```
Console.WriteLine("Division of 2 numbers is {0}", c)
```

2. Code: Use of unassigned variable

```
c = a / b;
Console.WriteLine("Division of 2 numbers is {0}", c);
```

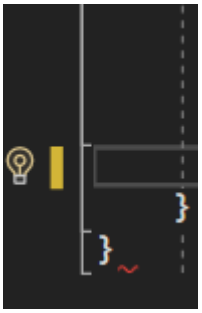
3. Code: Mistake in code

```
Console.Writeline("enter a:");
```

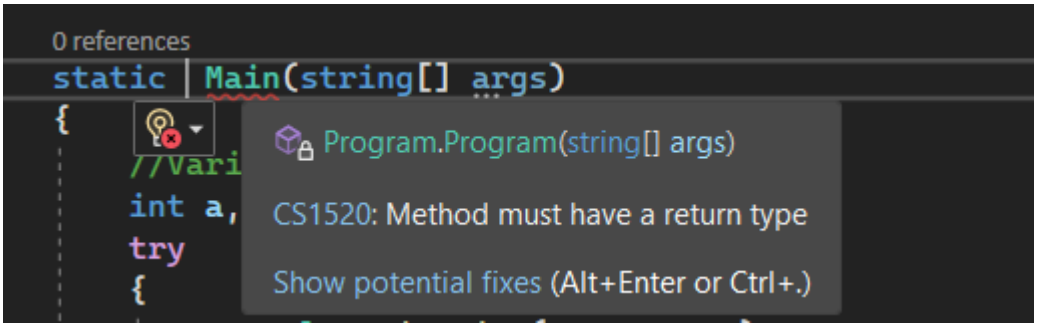
4. Code: Forget to implement namespace

```
Console.WriteLine("enter a:");
a = Convert.ToInt32(Console.ReadLine());
Console.WriteLine("enter b:");
b = Convert.ToInt32(Console.ReadLine());
```

5. Code:

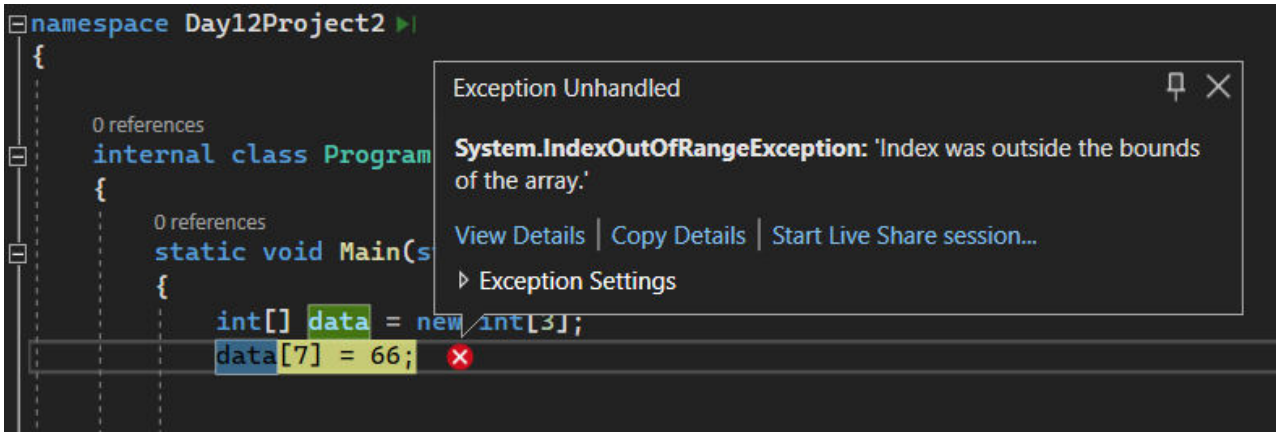


6. Code:

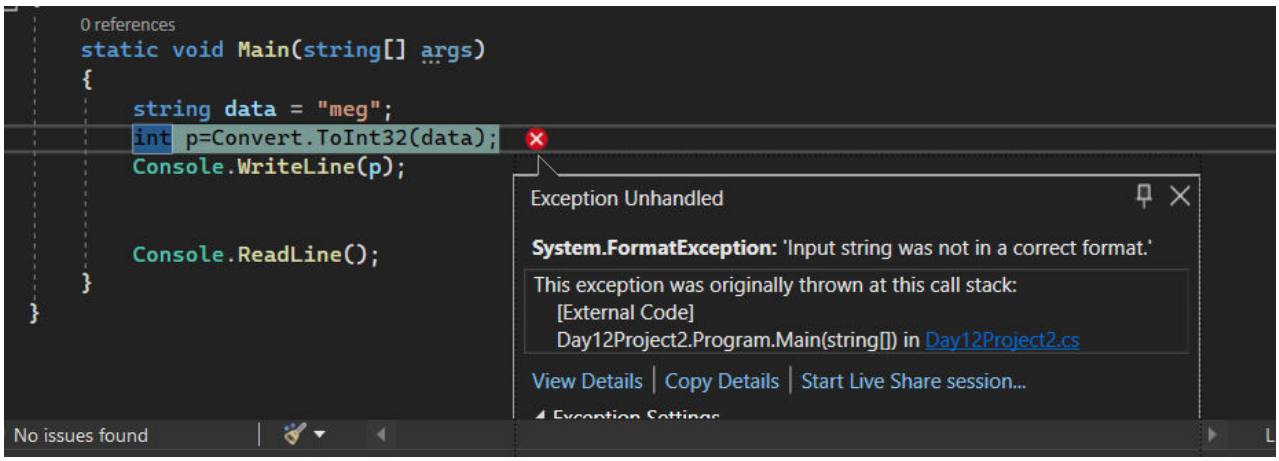


8.Write 6 runtime errors with small snippets.

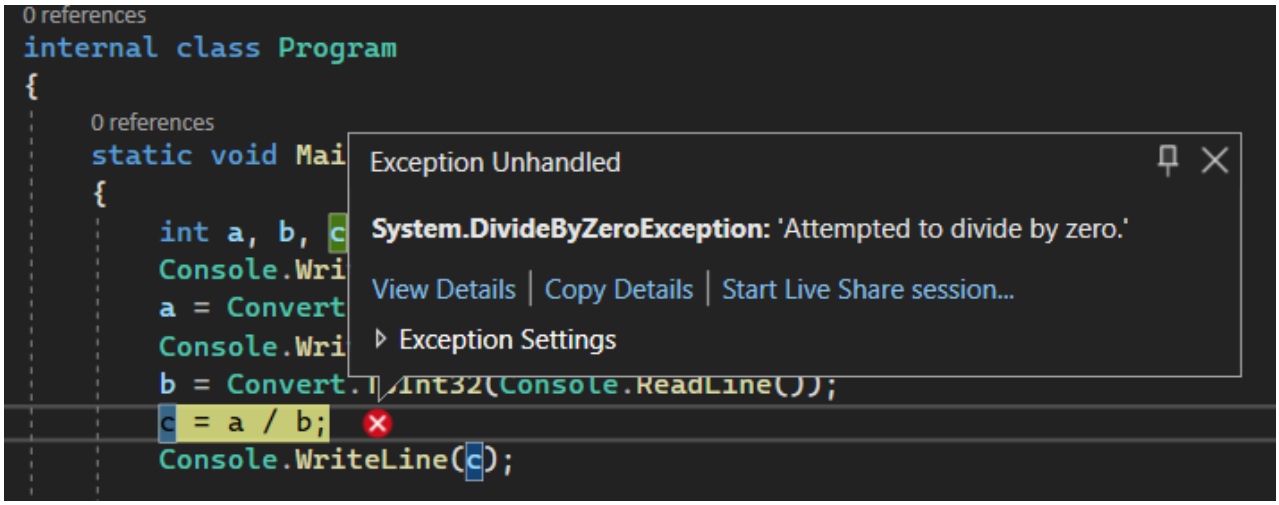
1. IndexOutOfRange



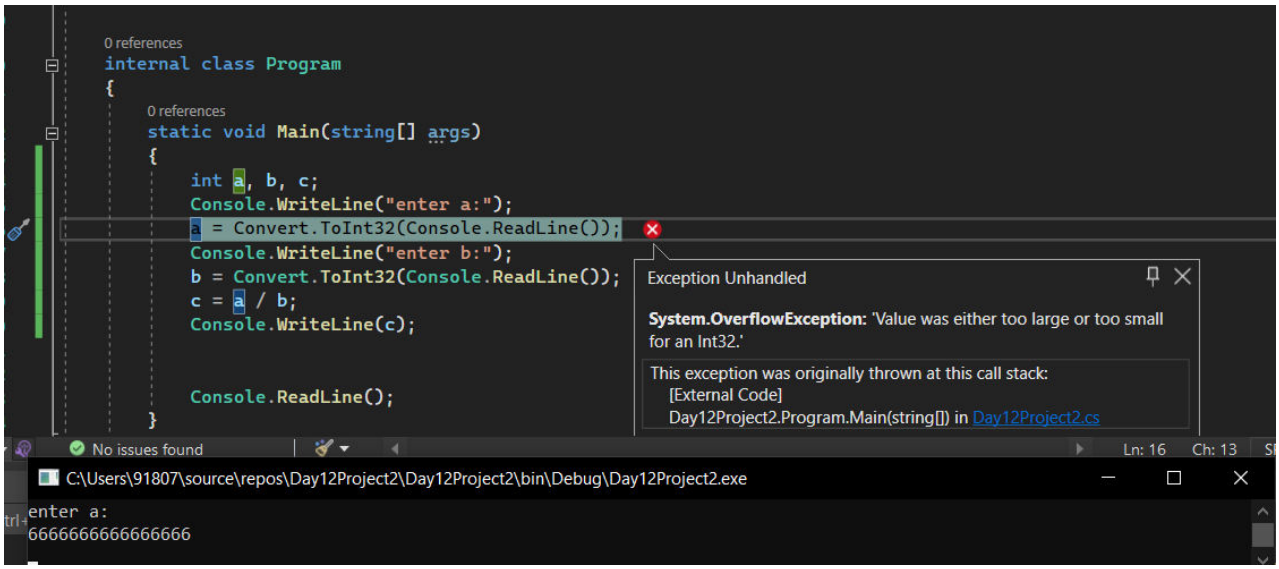
2. FormatException



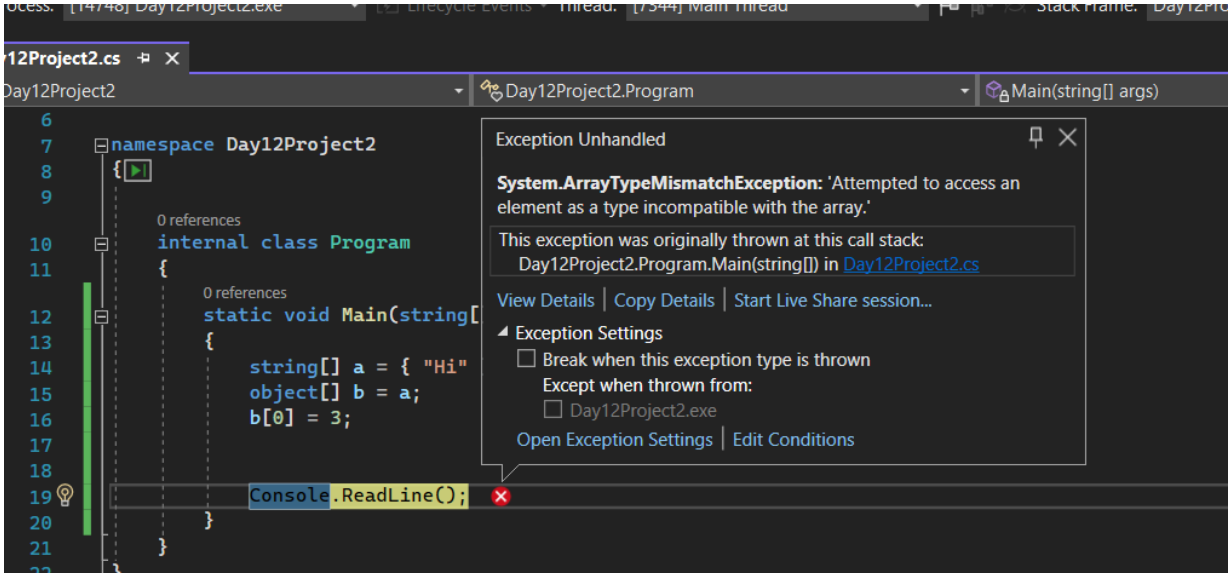
3. DivideByZero



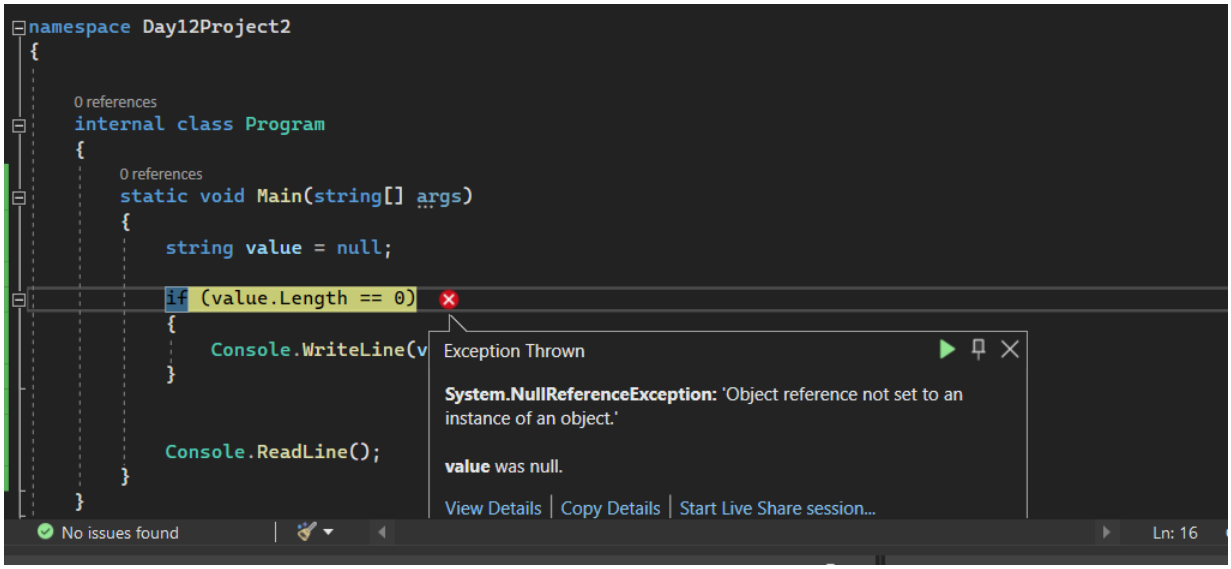
4. Overflow



5. ArrayTypeMisMatch



6. NullReference error:


namespace Day12Project2
{
0 references
internal class Program
{
0 references
static void Main(string[] args)
{
string value = null;
if (value.Length == 0)
{
Console.WriteLine(v
}
Console.ReadLine();
}
}
Exception Thrown
System.NullReferenceException: 'Object reference not set to an instance of an object.'
value was null.
View Details | Copy Details | Start Live Share session...
No issues found
Ln: 16

END OF DAY