NB Healthcare Technologies Pvt Ltd

Day 12 Morning Assignment (08 – Feb- 2022)

By

Vamsi Krishna Mandapati

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

Exception Handling :

Exception Handling is done to ensure that our application will not crash (or) will not display any technical details and to make sure we handle errors gracefully(properly) and display friendly messages.

Exception handling is important because **it helps maintain the normal, desired flow of the program even when unexpected events occur**. If exceptions are not handled, programs may crash or requests may fail. This can be very frustrating for customers and if it happens repeatedly, you could lose those customers.

Syntax:

```
try {
   // statements causing exception
} catch( ExceptionName e1 ) {
   // error handling code
} catch( ExceptionName e2 ) {
   // error handling code
} catch( ExceptionName eN ) {
   // error handling code
} finally {
   // statements to be executed
}
```

- Try : Used to define a try block. This block holds the code that may throw an exception
- Catch : Used to define a catch block. This block catches the exception thrown by the try block
- Finally : Used to define the finally block. This block holds the default code.
- Throw: used to throw an exception manually.

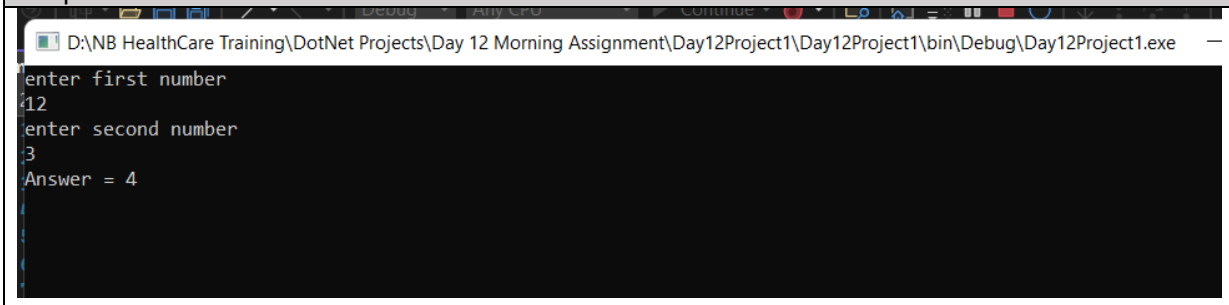
2. Write a simple division program and handle three exceptions discussed in the class., also add super exception at the last.

Code:

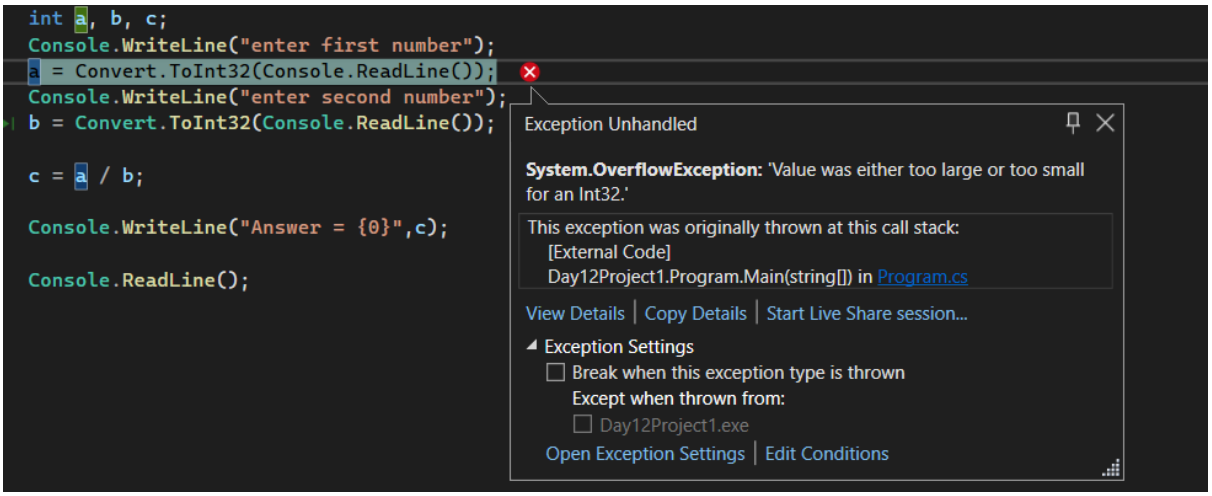
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Day12Project1
{

    internal class Program
    {
        static void Main(string[] args)
        {
            try
            {
                int a, b, c;
                Console.WriteLine("enter first number");
                a = Convert.ToInt32(Console.ReadLine());
                Console.WriteLine("enter second number");
                b = Convert.ToInt32(Console.ReadLine());

                c = a / b;

                Console.WriteLine("Answer = {0}", c);

                Console.ReadLine();

            }
            catch(OverflowException)
            {
                Console.WriteLine("only numbers between 0 and 500000 are
allowed");
                Console.ReadLine();
            }
            catch(DivideByZeroException)
            {
                Console.WriteLine("cannot divide with zero");
                Console.ReadLine();
            }
            catch(FormatException)
            {
                Console.WriteLine("only numbers are entered,please double
check");
                Console.ReadLine();
            }
            catch(Exception)
            {
                Console.WriteLine("some error occured.contact
admin@mycompany.com");
                Console.ReadLine();
```

}

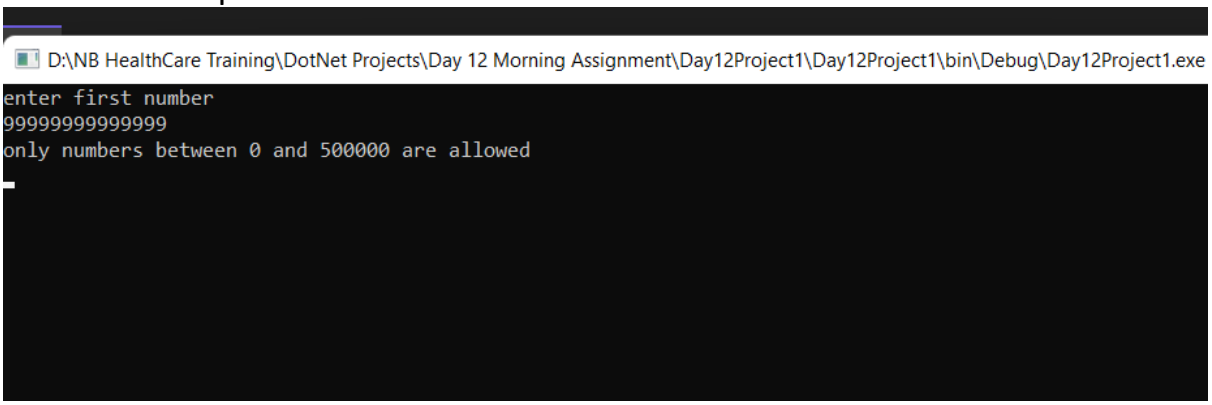
Output:



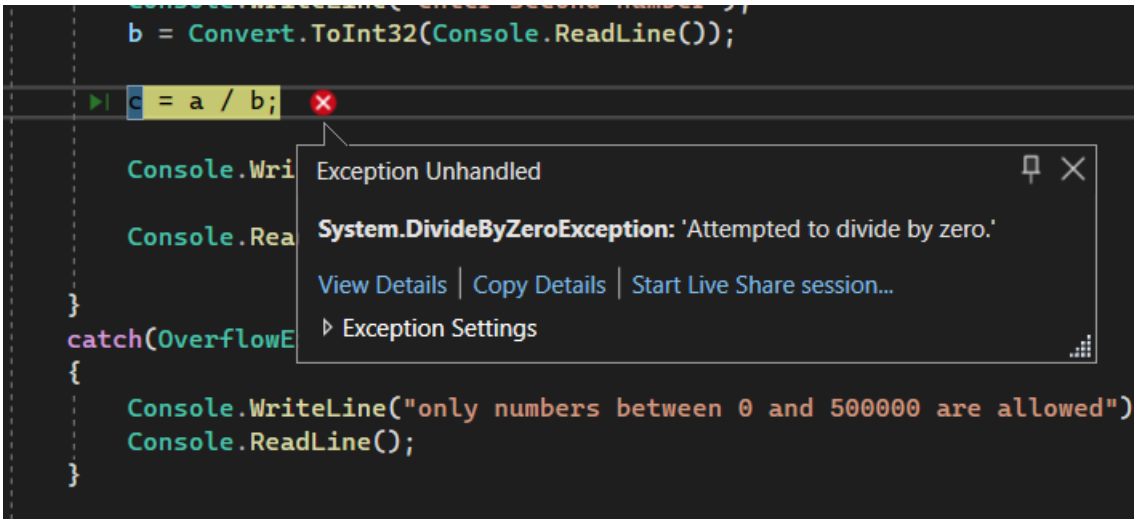
- When I enter 99999999999999999999 in the command prompt for first number, I got System.OverFlowException as shown in the below pic



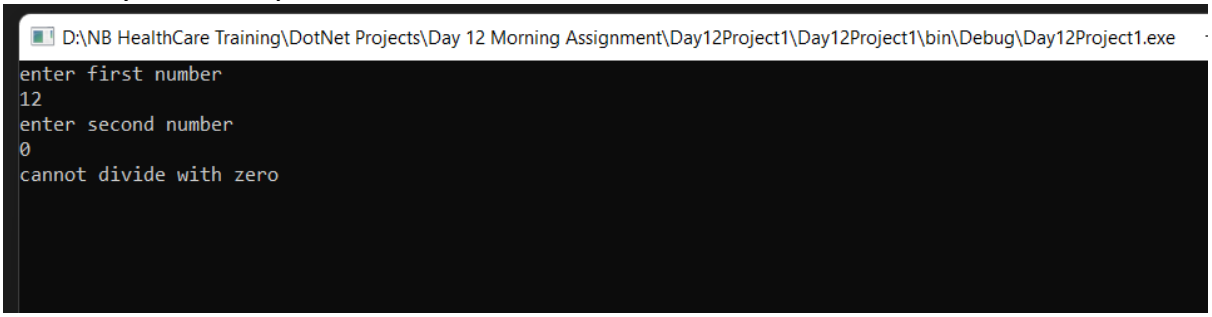
OverFlowException Handled:



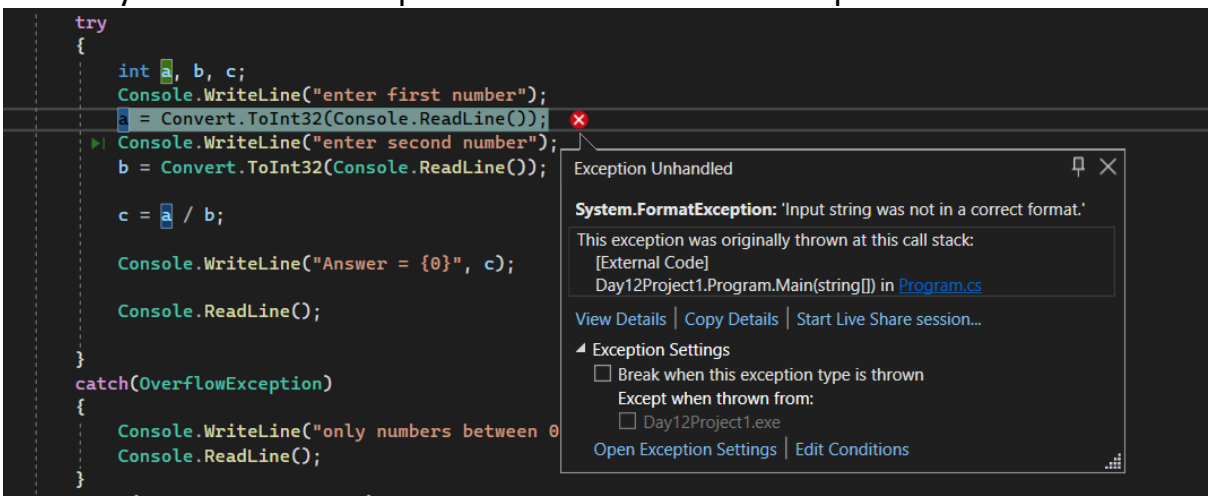
- When I enter 0 in the command prompt for second number, I got System.DivideByZeroException as shown in the below pic



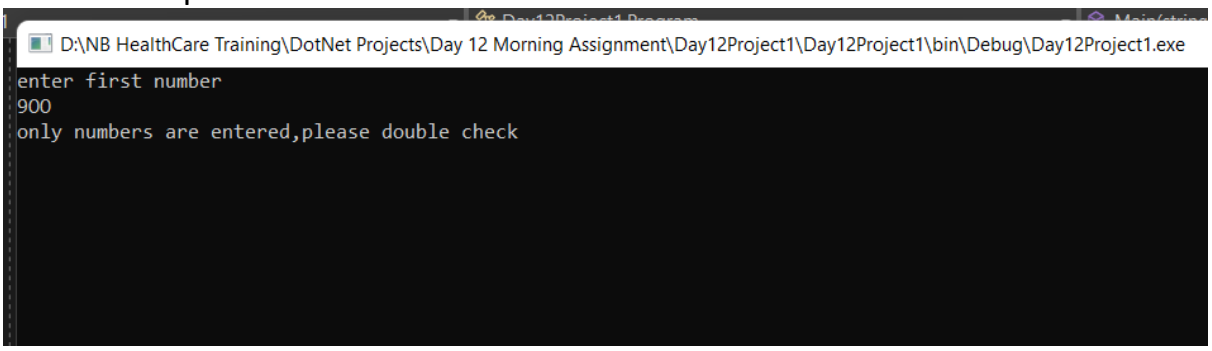
DivideByZeroException Handled:



- When I enter 9OO in the command prompt for first number, I got System.FormatException as shown in the below pic



FormatException Handled.



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

==========

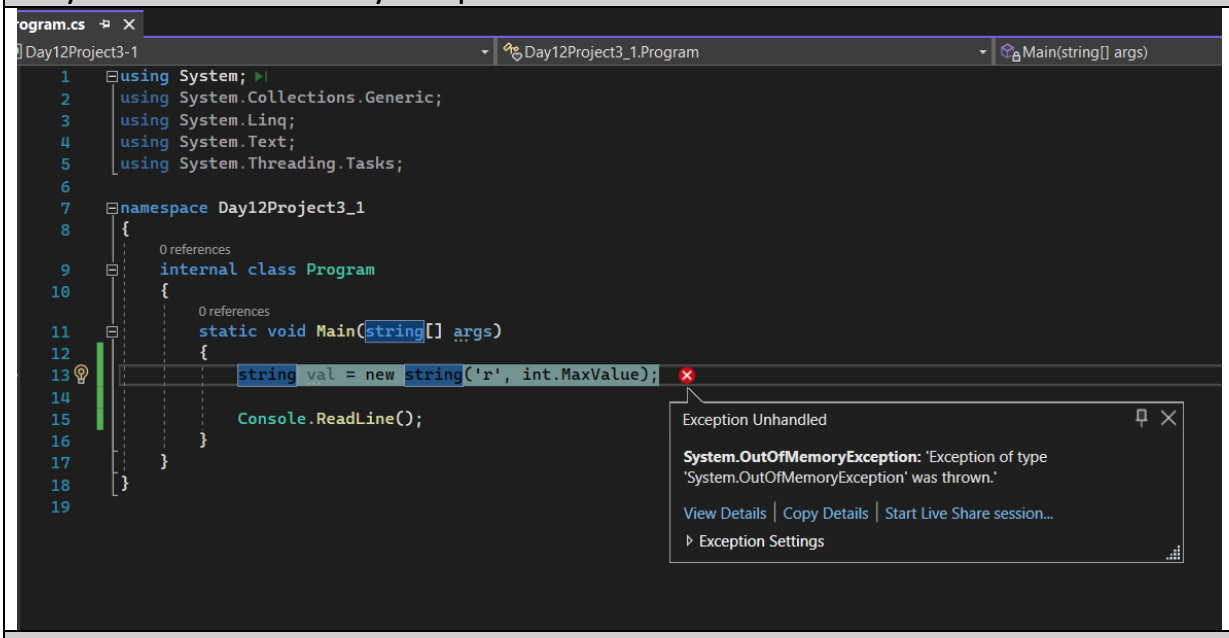
Divide by zero exception:

Reason : why you try to divide with zero.

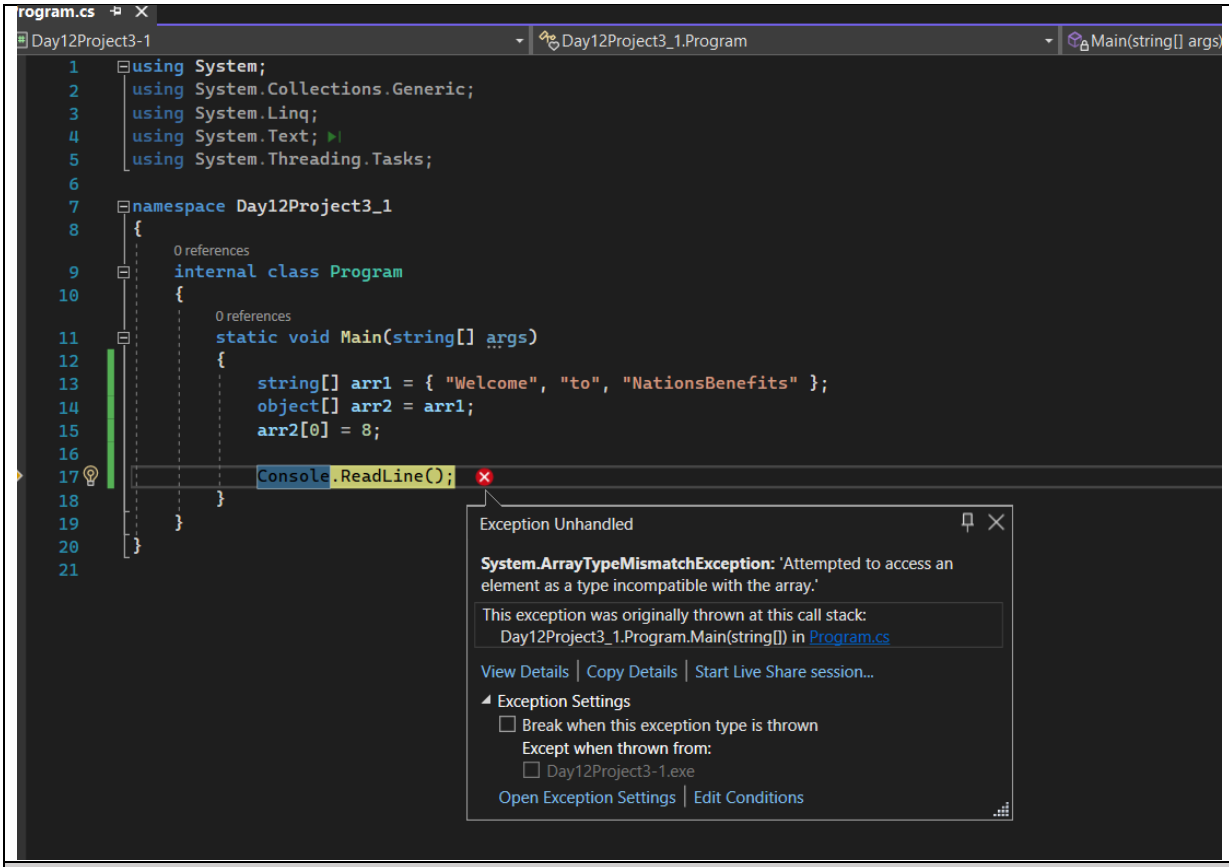
Example code:

```
int a=5;
int b=0;
int c=a/b;
```

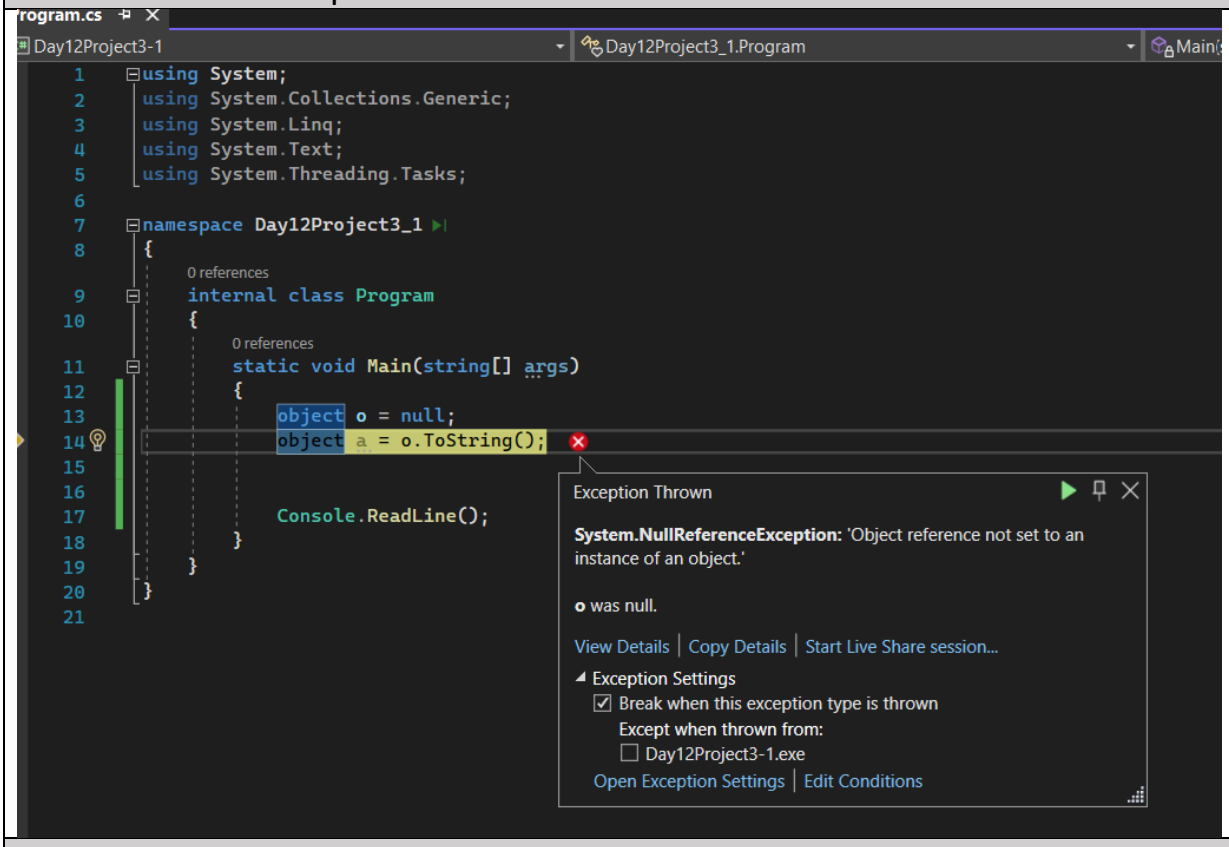
1.System.OutOfMemoryException



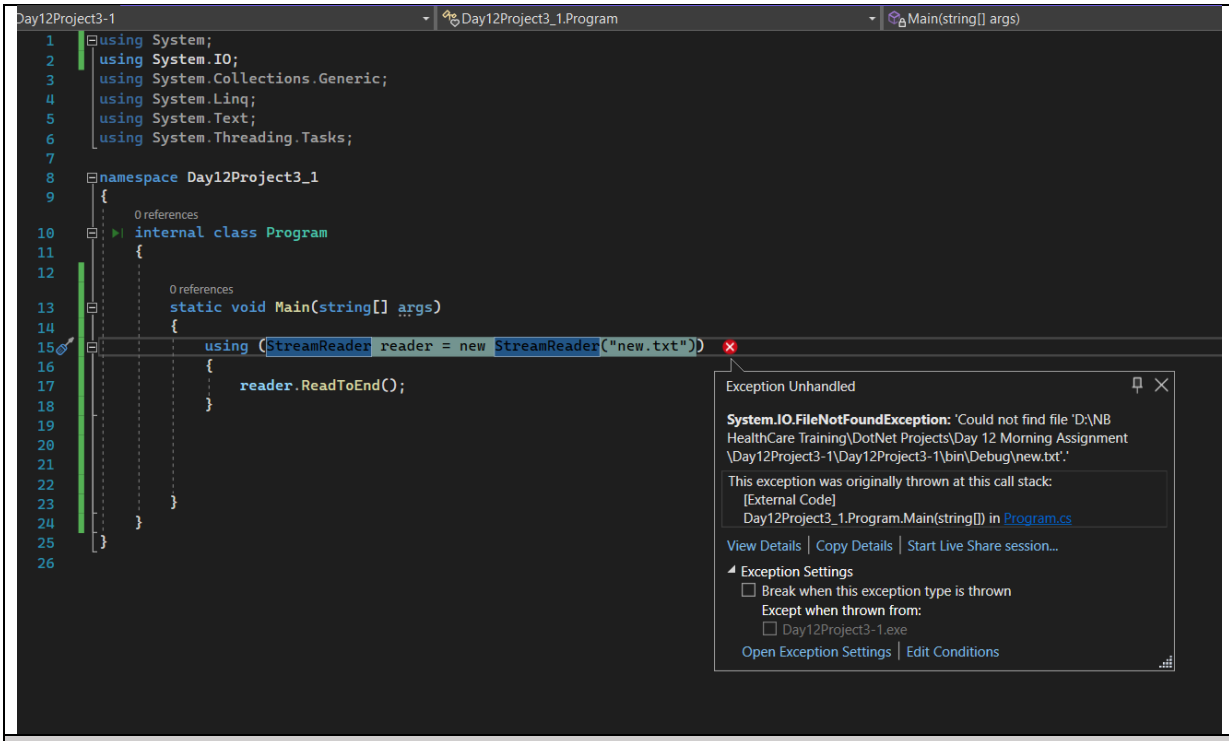
2.System.ArrayTypeMismatchException



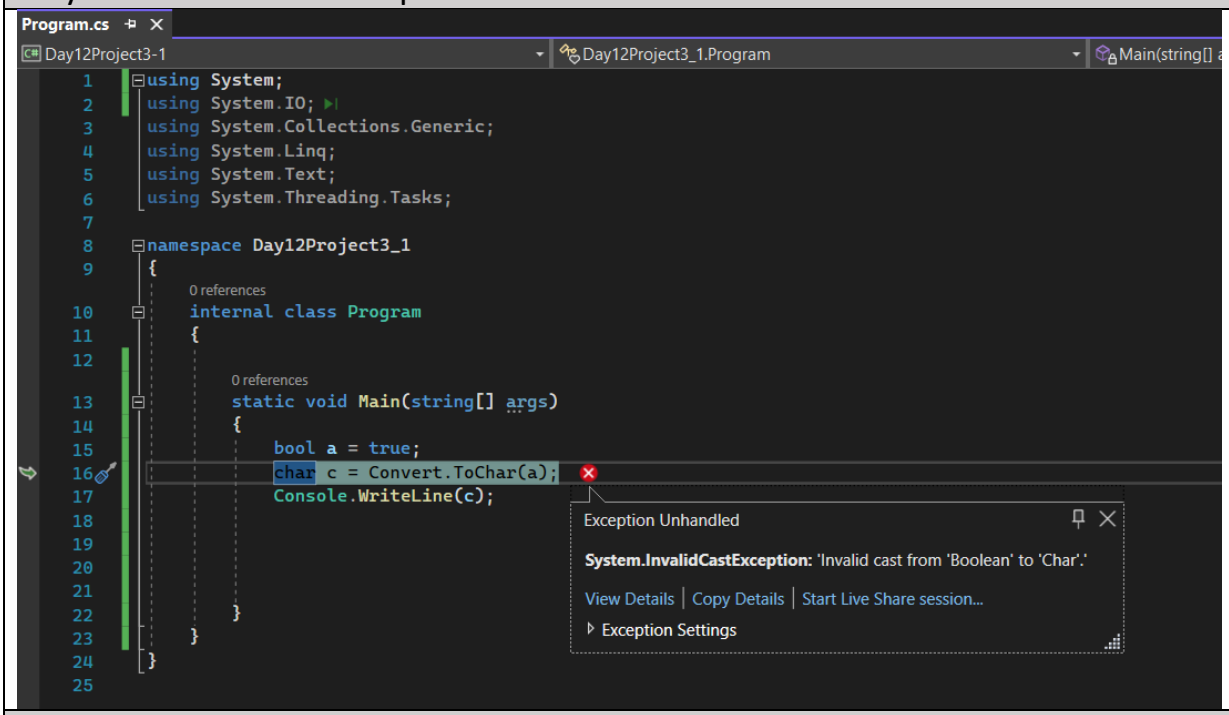
3.NullReferenceException



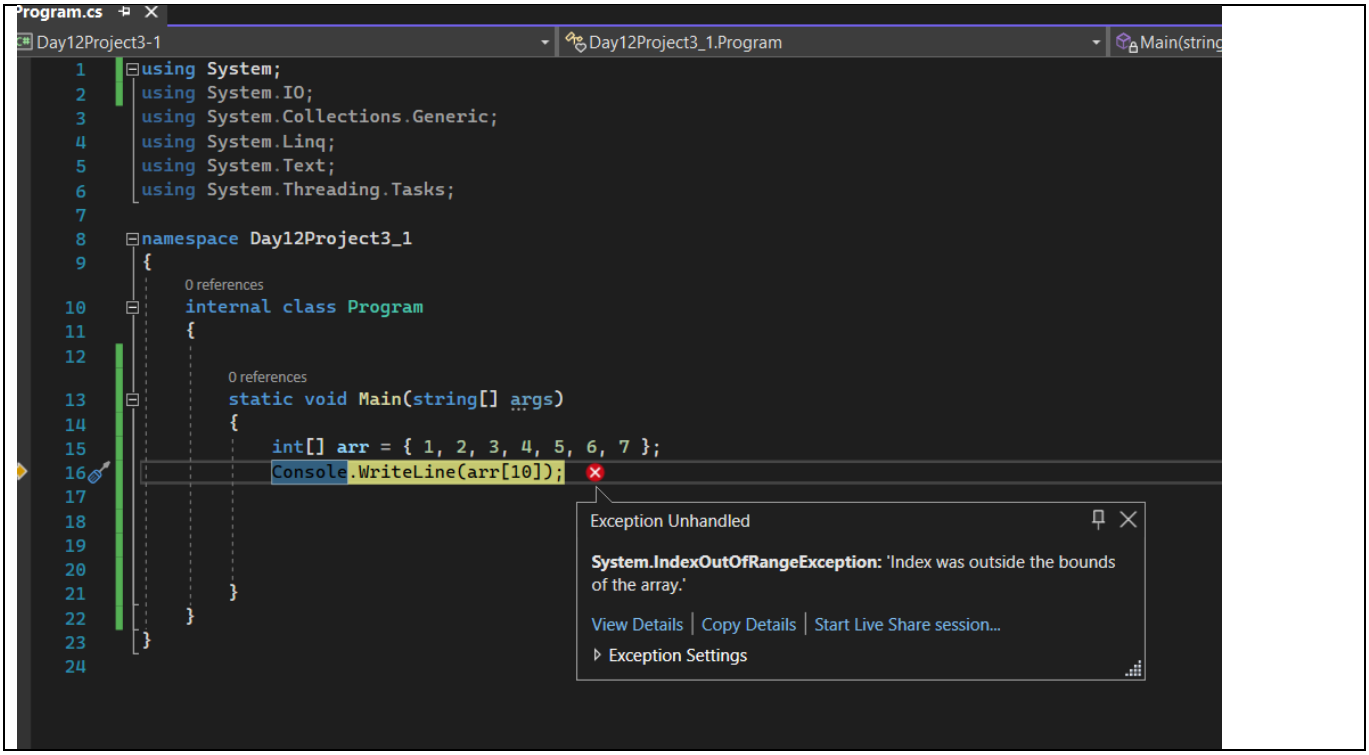
4.System.IO.FileNotFoundException



5.System.InvalidCastException



6.System.IndexOutOfRangeException



4. What is the use of "finally" block illustrate with an example.

Finally:

Statements Will Execute inside of finally block, irrespective of whether the exception occurs (or) not.

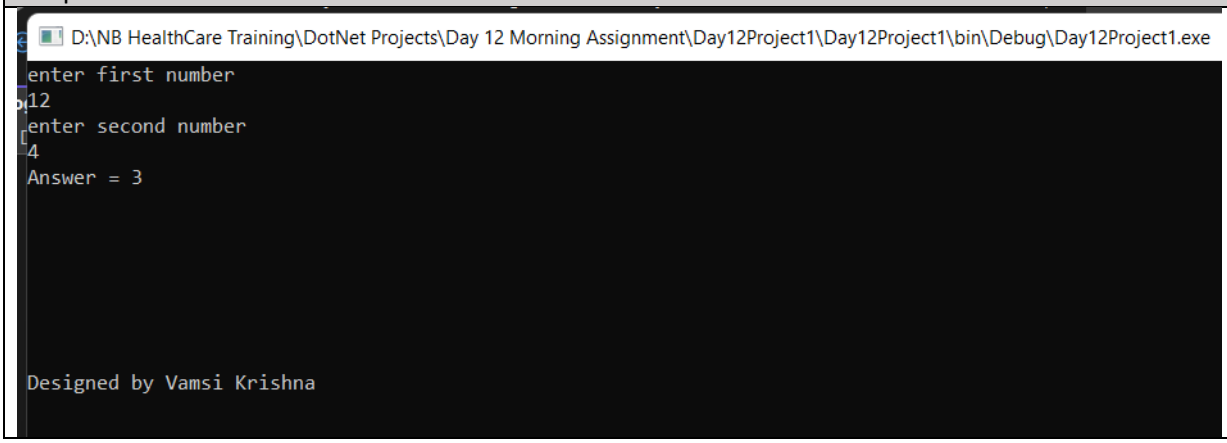
- **Sometimes there is a need to execute a set of code every time the program runs**. Even if the exception occurs and even if it doesn't, there can be some code that must be executed at end of the program. That code is written in finally block. This block is always executed regardless of exceptions occurring.
- Using a finally block,
  a) To display some common message.
  b) To close any opened database connection or any opened file connection
  c) you can clean up any resources that are allocated in a try block.

Code:

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

namespace Day12Project1
{

    internal class Program
```

```
    {
        static void Main(string[] args)
        {

                int a, b, c;
                Console.WriteLine("enter first number");
                a = Convert.ToInt32(Console.ReadLine());
                Console.WriteLine("enter second number");
                b = Convert.ToInt32(Console.ReadLine());

                c = a / b;

                Console.WriteLine("Answer = {0}", c);

                Console.ReadLine();



            finally
            {
                Console.WriteLine("\n\n\n\n\n\nDesigned by Vamsi Krishna");
                Console.ReadLine();
            }

        }
    }
}
```

Output:



5. Write the 5 points I explained about exception handling.

Exception Handling:

1.Exception handling is done to handle errors/exceptions gracefully, so that the application will not crash and without displaying any technical errors to the end customer/user.

2. A single try block can have multiple catch blocks.

3. always write the general exceptions at the last, if we write at the top it will handle all the exceptions so remaining exceptions will give error.

4. Statements Will Execute inside of finally block, irrespective of whether the exception occurs (or) not.

Or

Statements inside of finally block will execute all the times.

5. the general syntax/flow for writing exception is try block, catch block, finally block.

6.What is Compilation error and Runtime error .write atleast 3 differences between them.

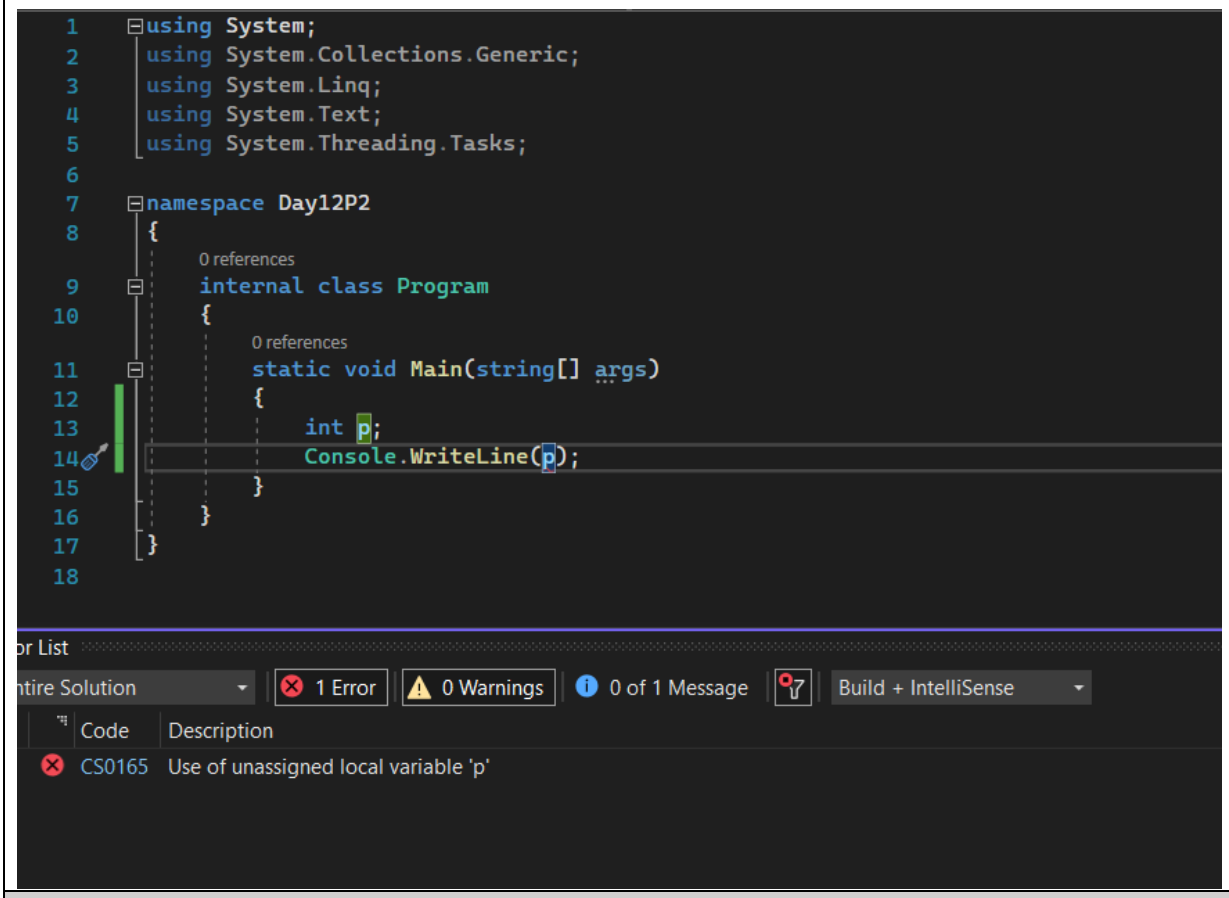
Compilation Error	Runtime Error
1.compilation error will occur when you write any spelling mistakes, wrong syntax in the code. A compiler error **happens when you try to compile the code**. If you are unable to compile your code, that is a compiler error.	1.Even though our code has no compilation errors ,but we may get runtime errors. A runtime error happens during the running of the program.
2. compilation errors will shown at the time of writing the code in the visual studio or in compilation time • Easily we can know compilation errors in visual studio by Redline. • To find the compilation errors, build the code.	2. Run time errors are shown at the code execution time or Code Runtime. If you compile and run your code, but then it fails during execution, that is runtime.

3.Use of Unassigned local Variables.
Ex: public static void Main(string[] args)
{
int p;
Console.WriteLine(p);
}
	3. A runtime error causes the program to terminate abnormally during execution of a program.

4. Compilation Error will occur if we did not import respective namespace.	4.Runtime errors will occur due to wrong logic in the code

7. Write any 6 compilation errors with small code snippet. Add compilation error screen shots.

1. Use of unassigned local variable p



2. ; missing at 14th line ending.

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Day12P2
8  {
       0 references
9      internal class Program
10     {
           0 references
11         static void Main(string[] args)
12         {
13             int p = 0;
14             Console.WriteLine(p)
15         }
16     }
17 }
18
```

```
rror List
Entire Solution | 1 Error | 0 Warnings | 0 Messages | Build + IntelliSense
Code | Description
CS1002 ; expected
```

3. Missing namespace

```
1  //using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Day12P2
8  {
       0 references
9      internal class Program
10     {
           0 references
11         static void Main(string[] args)
12         {
13             int p = 0;
14             Console.WriteLine(p);
15         }
16     }
17 }
18
```

```
or List
ntire Solution | 1 Error | 0 Warnings | 0 Messages | Build + IntelliSense
Code | Description
CS0103 The name 'Console' does not exist in the current context
```

4.Wrong datatype

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Day12P2
8  {
       0 references
9      internal class Program
10     {
           0 references
11         static void Main(string[] args)
12         {
13             string p = 0;
14             Console.WriteLine(p);
15         }
16     }
17 }
18
```

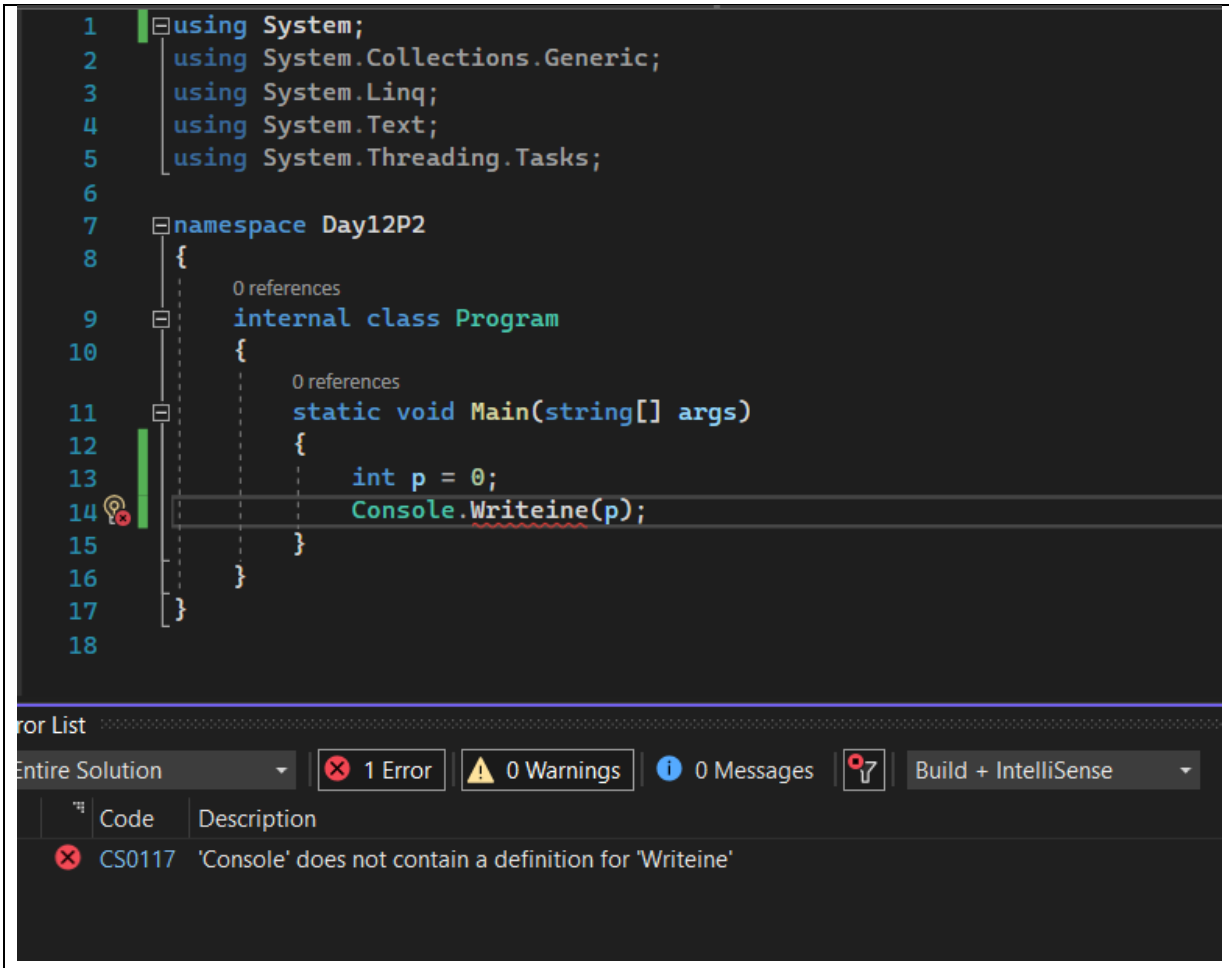
rror List

Entire Solution | 1 Error | 0 Warnings | 0 of 1 Message | Build + IntelliSense

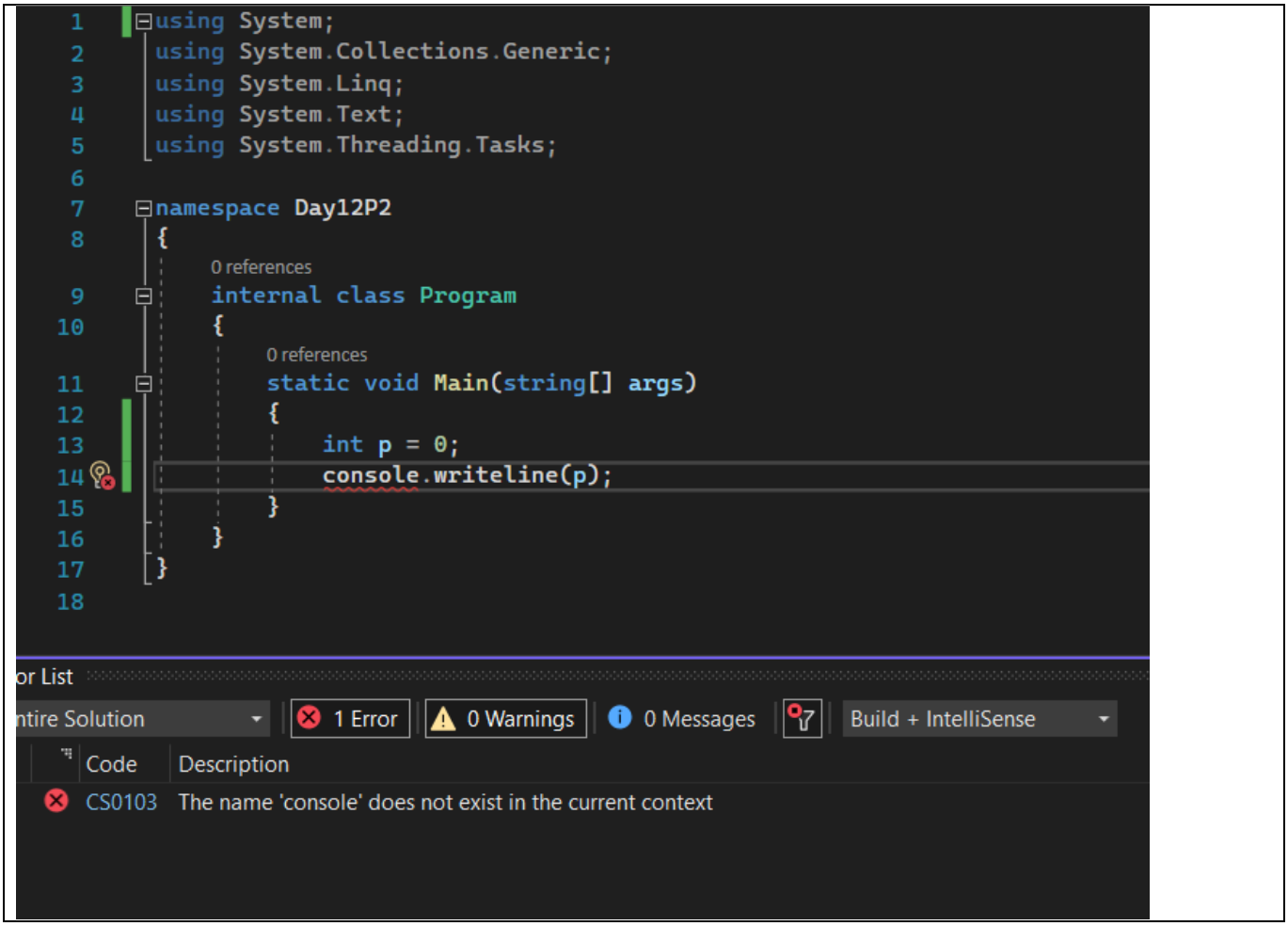
1 Error

Code	Description
CS0029	Cannot implicitly convert type 'int' to 'string'

5.Spelling mistake



6.Not following naming conventions i.e case sensitive 14th line



8. Write any 6 runtime errors with small code snippets and add run time error screen shots.

1.System.OverflowError

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

namespace Day12P2
{
    0 references
    internal class Program
    {
        0 references
        static void Main(string[] args)
        {
            int a, b, c;
            Console.WriteLine("enter first number");
            a = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("enter second number");
            b = Convert.ToInt32(Console.ReadLine());

            c = a + b;

            Console.WriteLine("Answer = {0}", c);

            Console.ReadLine();


        }
    }
}
```

Exception Unhandled

System.OverflowException: 'Value was either too large or too small for an Int32.'

This exception was originally thrown at this call stack:
[External Code]
Day12P2.Program.Main(string[]) in Program.cs

View Details | Copy Details | Start Live Share session...

Exception Settings
Break when this exception type is thrown
Except when thrown from:
Day12P2.exe

Open Exception Settings | Edit Conditions

2. System.FormatException Error

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

namespace Day12P2
{
    0 references
    internal class Program
    {
        0 references
        static void Main(string[] args)
        {
            string data = "Vamsi";

            int p = Convert.ToInt32(data);

            Console.WriteLine(p);

            Console.ReadLine();
        }
    }
}
```

Exception Unhandled

System.FormatException: 'Input string was not in a correct format.'

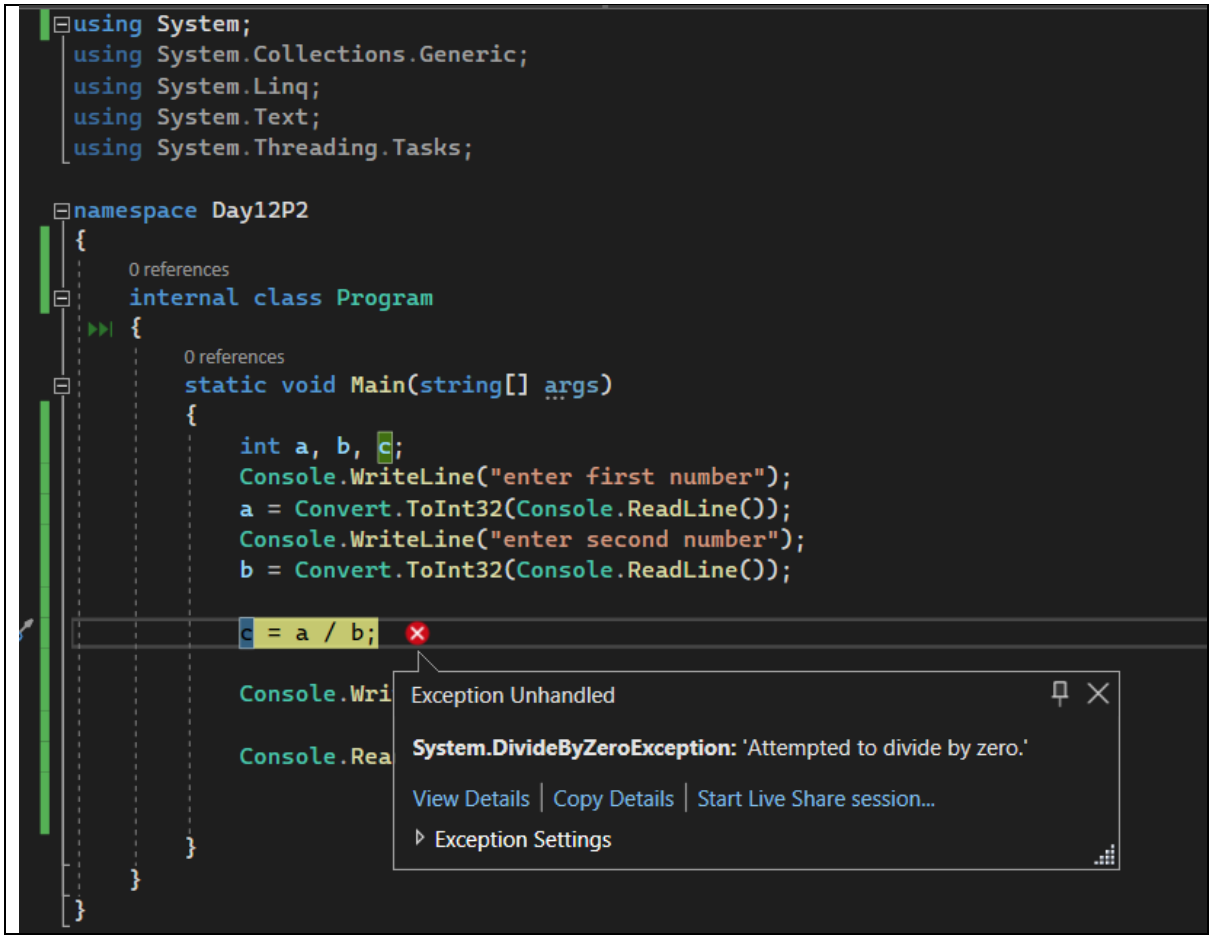
This exception was originally thrown at this call stack:
[External Code]
Day12P2.Program.Main(string[]) in Program.cs

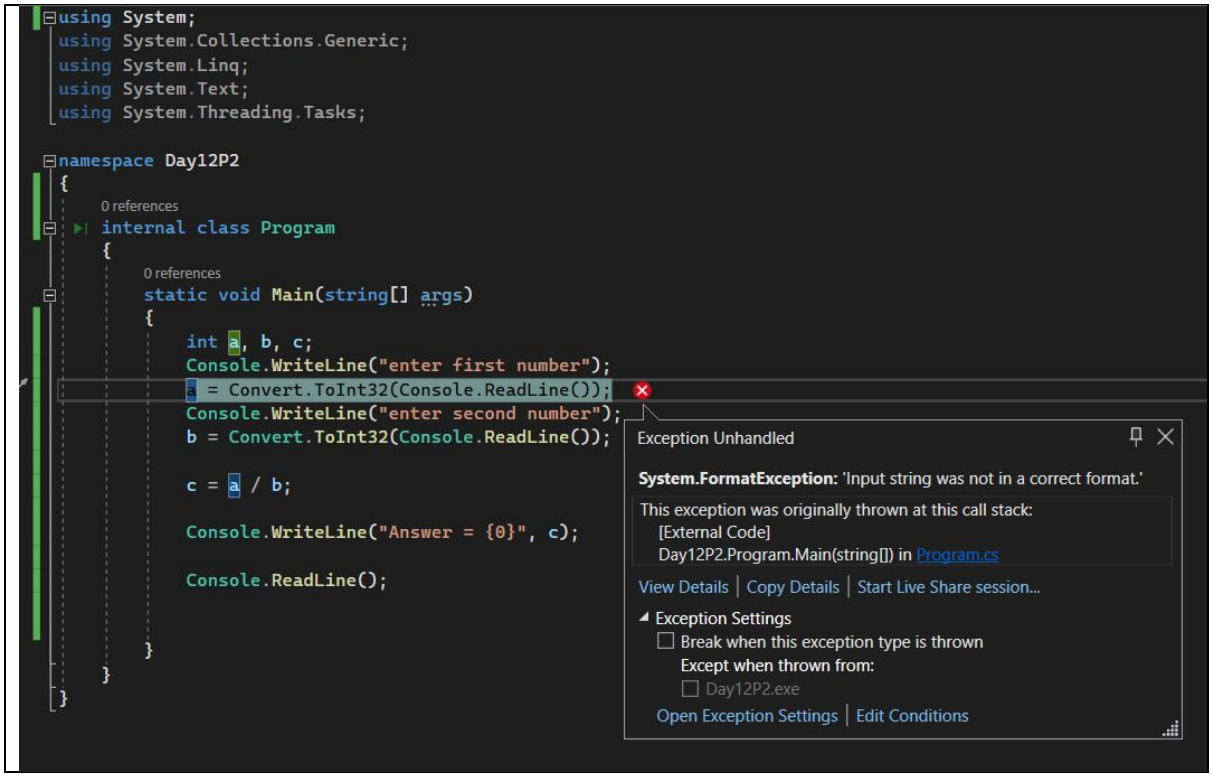
View Details | Copy Details | Start Live Share session...

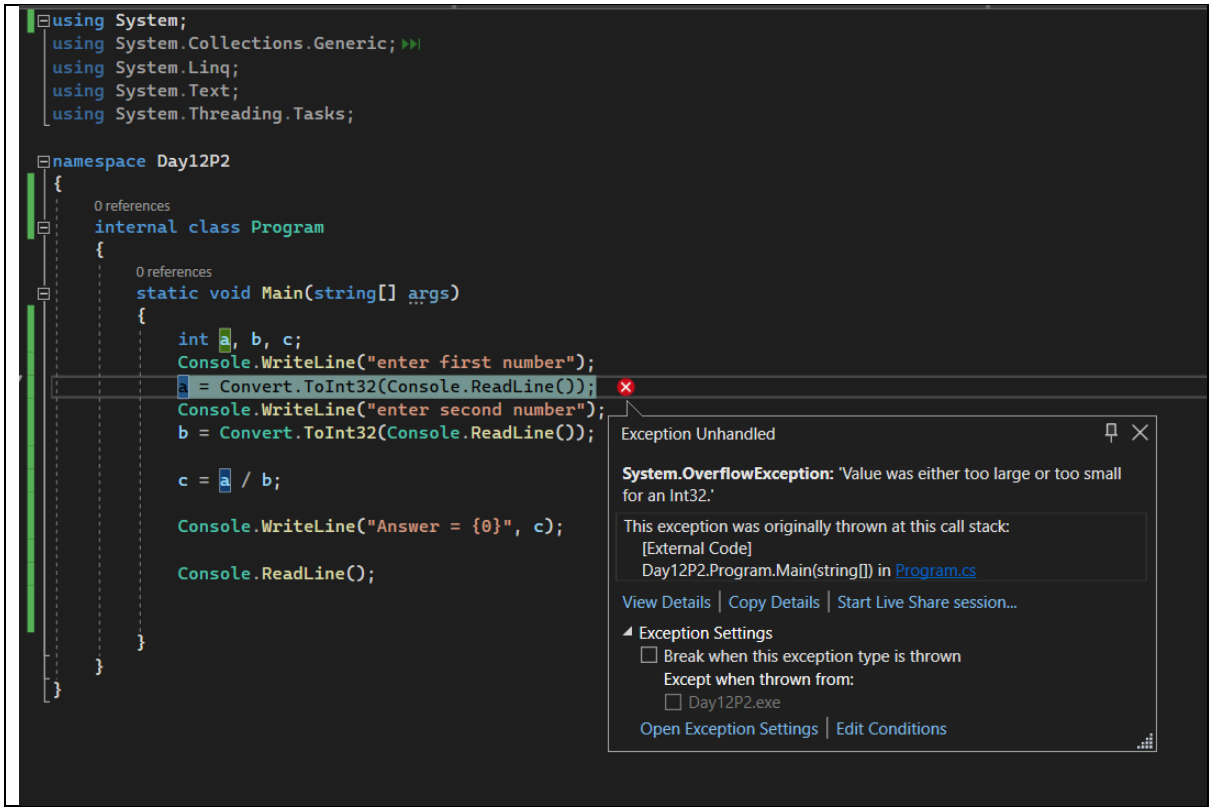
Exception Settings
Break when this exception type is thrown
Except when thrown from:
Day12P2.exe

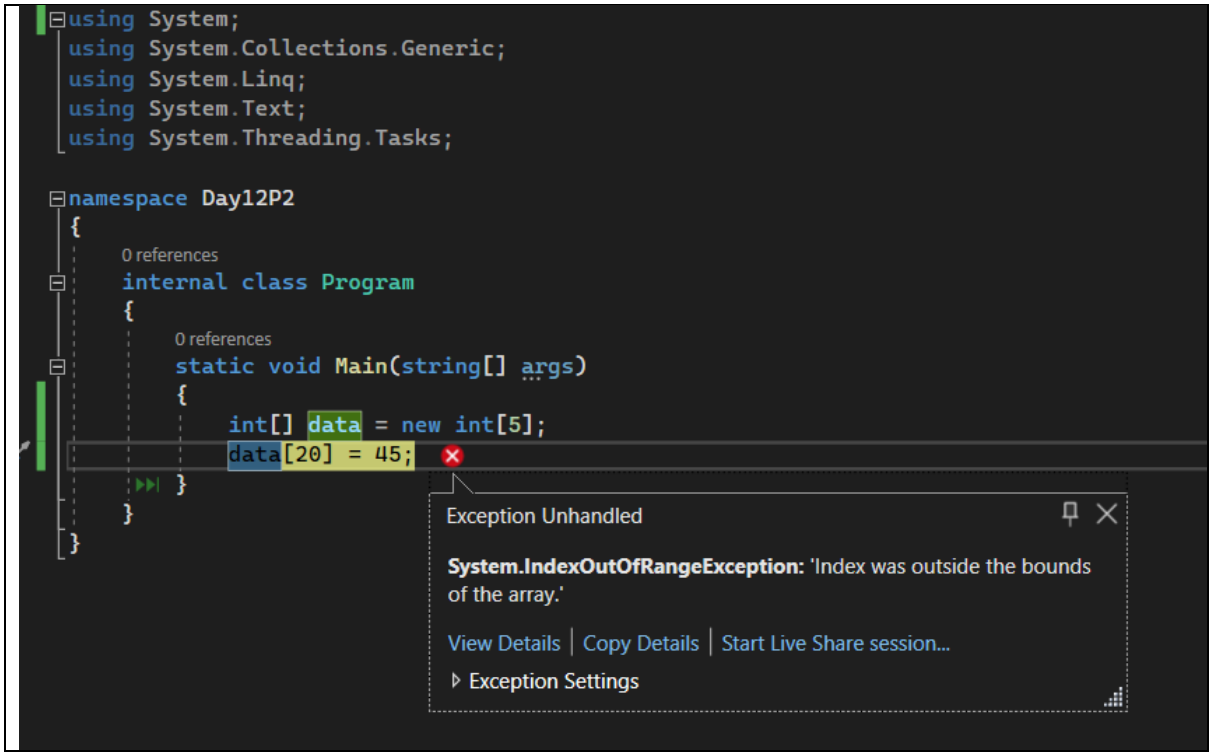
Open Exception Settings | Edit Conditions

3.System.DivideByZeroException Error


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace Day12P2
{
0 references
internal class Program
{
0 references
static void Main(string[] args)
{
int a, b, c;
Console.WriteLine("enter first number");
a = Convert.ToInt32(Console.ReadLine());
Console.WriteLine("enter second number");
b = Convert.ToInt32(Console.ReadLine());
c = a / b;
Console.Wri
Console.Rea
}
}
}
Exception Unhandled
System.DivideByZeroException: 'Attempted to divide by zero.'
View Details | Copy Details | Start Live Share session...
Exception Settings


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace Day12P2
{
0 references
internal class Program
{
0 references
static void Main(string[] args)
{
int a, b, c;
Console.WriteLine("enter first number");
a = Convert.ToInt32(Console.ReadLine());
Console.WriteLine("enter second number");
b = Convert.ToInt32(Console.ReadLine());
c = a / b;
Console.WriteLine("Answer = {0}", c);
Console.ReadLine();
}
}
}
Exception Unhandled
System.FormatException: 'Input string was not in a correct format.'
This exception was originally thrown at this call stack:
[External Code]
Day12P2.Program.Main(string[]) in Program.cs
View Details | Copy Details | Start Live Share session...
Exception Settings
Break when this exception type is thrown
Except when thrown from:
Day12P2.exe
Open Exception Settings | Edit Conditions


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace Day12P2
{
0 references
internal class Program
{
0 references
static void Main(string[] args)
{
int a, b, c;
Console.WriteLine("enter first number");
a = Convert.ToInt32(Console.ReadLine());
Console.WriteLine("enter second number");
b = Convert.ToInt32(Console.ReadLine());
c = a / b;
Console.WriteLine("Answer = {0}", c);
Console.ReadLine();
}
}
}
Exception Unhandled
System.OverflowException: 'Value was either too large or too small for an Int32.'
This exception was originally thrown at this call stack:
[External Code]
Day12P2.Program.Main(string[]) in Program.cs
View Details | Copy Details | Start Live Share session...
Exception Settings
Break when this exception type is thrown
Except when thrown from:
Day12P2.exe
Open Exception Settings | Edit Conditions


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace Day12P2
{
0 references
internal class Program
{
0 references
static void Main(string[] args)
{
int[] data = new int[5];
data[20] = 45;
}
}
}
Exception Unhandled
System.IndexOutOfRangeException: 'Index was outside the bounds of the array.'
View Details | Copy Details | Start Live Share session...
Exception Settings