R·I·T	Rochester Institute of Technology Golisano College of Computing and Information Sciences Interactive Games and Media Department

Unit Test #1

1. Create a console application that enhances the attached "Math Quiz Application" to include division. Note that you will have to replace the **int** variables with **doubles** and you should round the quotient to 2 decimal places and indicate that the user should enter their answer to 2 decimal places. (dAnswer = Math.Round(dAnswer,2))

   GitHub URL: https://github.com/Sebas11an245/myIGME-201/tree/main/Unit%20Test%20%231/MathQuiz

2. Create a console application that modifies the attached "Number Sorter" application to request sentences and sort the words in the sentence in ascending or descending alphabetical order.

   Note that you cannot compare strings by using < or >.
   You will have to use string1.CompareTo(string2) which:
   - returns a negative number if string1 < string2
   - returns a positive number if string1 > string2

   GitHub URL: https://github.com/Sebas11an245/myIGME-201/tree/main/Unit%20Test%20%231/String%20Sorter

3. Create a console application that uses a delegate to impersonate the `Math.Round(double, int)` function. (Refer to "Math Delegate", "MemoryGame" or the attached "Number Sorter" application for delegate code examples). 1 extra point will be given for each additional implementation you can demonstrate using abbreviated notation, anonymous methods, the lambda operator and/or the generic template type. (Up to 5 extra points are available!)

   GitHub URL: https://github.com/Sebas11an245/myIGME-201/tree/main/Unit%20Test%20%231/MathRoundImpersonation

4. Write a console application that re-creates the attached 3questions.exe.

   GitHub URL: https://github.com/Sebas11an245/myIGME-201/tree/main/Unit%20Test%20%231/ThreeQuestions

5. Which of the following conversions can't be performed implicitly:

```
a. ulong to int
     a. ulong is larger than int
b. uint to long
c. bool to string
     a. bool has no implicit conversion to string
d. float to byte

     a. float is a floating point value and byte isnt
```

6. Given the following namespaces:

```
namespace Worst
{
    using When = Episode;
```

```
        // name "Meme" defined

        namespace Episode
        {
                // name "Ever" defined
        }
}

// global code
```

a. How would you reference "Ever" from the global code?
    a. Worst.Episode.Ever
b. How would you reference "Ever" from the "Worst" namespace using the alias?
    a. Worst.When.Ever
c. How would you reference "Ever" from the "Episode" namespace?
    a. Worst.Episode.Ever
d. How would you reference "Meme" from the global code?
    a. Worst.Meme
e. How would you reference "Meme" from the "Worst" namespace?
    a. Worst.Meme
f. How would you reference "Meme" from the "Episode" namespace?
    a. Worst.When.Meme

7. By considering operator precedence, list the steps involved in the computation of the following expression:

```
resultVar -= var1 + var2 * var3 % var4 - var5;
```

```
1.var2 * var3
2.(var2 * var3) % var4
3.var1 + ((var2 * var3) % var4)
4.resultVar -= (var1 + ((var2 * var3) % var4)) - var5
```

8. Given the formula $z = 4y^3 + 2x^2 - 8x + 7$ implement a multidimensional array, the necessary for() loops and the code to store the values of z for the following ranges of x and y into the array:
    - -1 <= y <= 1 in 0.1 increments
    - 0 <= x <= 4 in 0.1 increments.

Use the delegate method you wrote in #3 to round x and y to 1 decimal, and z to 3 decimals

```
RoundDelegate roundFunction = Math.Round;
// -1 <= x <= 1 in 0.1 increments: 21 values of x
// 1 <= y <= 4 in 0.1 increments: 31 values of y
```

```
// use the 3rd dimension to store each value of x,y,z
double[,,] zFunc = new double[21, 31, 3];
// our doubles to hold the values to compute with
double dX, dY, dZ;
// our ints to access the array (array indexes can only be integers)
int nX, nY;
// start nX array index at 0
nX = 0;
// loop through all values of x
for( dX = -1; dX <= 1; dX += 0.1 )
{
// doubles use base-2 calculations which introduce rounding
errors
// round each increment to 1 decimal place
dX = roundFunction(dX, 1);
// for each value of dX loop through all values of dY
// restart the nY array element index for each value of nX
nY = 0;
// loop through all values of y
for( dY = 1; dY <= 4; dY += 0.1 )
{
// round each increment to 1 decimal place
dY = roundFunction(dY, 1);
// calculate z
dZ = 4 * dY * dY * dY + 2 * dX *dX – 8 * dX + 7;
// round the calculation to 3 decimal places
dZ = roundFunction(dZ, 3);
// store x, y, and z for the current array element [nX,nY]
zFunc[nX, nY, 0] = dX;
zFunc[nX, nY, 1] = dY;
zFunc[nX, nY, 2] = dZ;
// increment the nY array index
++nY;
}
// increment the nX array index
++nX;
}
```

9. List all of the errors in the following code (Hint: there are 5 errors):

```
double[][] dArray = new int[2][];
dArray[1] = new double[2];
dArray[2] = new double[1]
dArray[0][0] = 15;
dArray[1][2] = 5.67;
```

1. It is trying to initialize a double array with int[][]

2. Lines 2 and 3 are accessing the 2nd and 3rd indexes of dArray instead of the 1st and 2nd
3. dArray[2] will result in an IndexOutOfRangeException
4. dArray[1][2] will result in an IndexOutOfRangeException
5. There is a missing semicolon on line 3

10. a. What will happen if the following code executes and why?

```
byte byteVal;
short shortVal = -556;
byteVal = (byte)shortVal;
Console.WriteLine("byteVal = {0}", byteVal);
```

The output will be byteVal = 76 because it had to shorten the value of the short to fit in a byte

b. What will happen if the following code executes and why?

```
byte byteVal;
short shortVal = -556;
byteVal = checked((byte)shortVal);
Console.WriteLine("byteVal = {0}", byteVal);
```

The code will throw an exception when it tries to convert from short to byte because of overflow.

c. What will happen if the following code executes and why?

```
byte byteVal;
short shortVal = -556;
byteVal = Convert.ToByte(shortVal);
Console.WriteLine("byteVal = {0}", byteVal);
```

The code will throw an exception when it tries to convert from short to byte because of overflow.

11. Which of the following are NOT legal C# identifier names?

$salary- Starts with '$'

MAX_INT_# - contains '#'

99bottlesOfBeerOnTheWall

#hashtag- contains/starts with '#'

excitement!- contains '!'

good&evil- contains '&'

12. Write a console application that has the following variable declarations:

```
string sName;
double dSalary = 30000;
```

It should prompt for the user's name, then call the following function:

```
static bool GiveRaise( string name, ref double salary )
```

The function should increase the salary by $19,999.99 if name = your name and return true

Otherwise it should return false.

The main program should congratulate the user if they got a raise, and display their new salary.

GitHub URL: https://github.com/Sebas11an245/myIGME-201/tree/main/Unit%20Test%20%231/SalaryRaise

13. Rewrite #12 using the following struct:

```
struct Employee
{
        public string sName;
        public double dSalary;
}
```

And rewrite the GiveRaise() function to pass the struct as the only parameter

GitHub URL: https://github.com/Sebas11an245/myIGME-201/tree/main/Unit%20Test%20%231/SalaryRaise

14. Create a "Console App (.NET Framework)" called "UT1_BugSquash". Paste the following code into the Program.cs file. Identify the compile-time, run-time and logical errors by commenting out the offending lines, explaining the type of error, and rewriting them correctly, or adding any additional code required to prevent any possible run-time or logical errors. Ensure the code is fixed to work as documented.

GitHub URL: https://github.com/Sebas11an245/myIGME-201/tree/main/Unit%20Test%20%231/UT1_BugSquash

```
using System;

namespace UT1_BugSquash
{
    class Program
    {
        // Calculate x^y for y > 0 using a recursive function
        static void Main(string[] args)
        {
            string sNumber;
            int nX;
            int nY
            int nAnswer;

            Console.WriteLine(This program calculates x^y.);

            do
            {
                Console.Write("Enter a whole number for x: ");
                Console.ReadLine();
            } while (!int.TryParse(sNumber, out nX));

            do
            {
                Console.Write("Enter a positive whole number for y: ");
```

```
            sNumber = Console.ReadLine();
        } while (int.TryParse(sNumber, out nX));

        // compute the exponent of the number using a recursive function
        nAnswer = Power(nX, nY);

        Console.WriteLine("{nX}^{nY} = {nAnswer}");
    }


    int Power(int nBase, int nExponent)
    {
        int returnVal = 0;
        int nextVal = 0;

        // the base case for exponents is 0 (x^0 = 1)
        if (nExponent == 0)
        {
            // return the base case and do not recurse
            returnVal = 0;
        }
        else
        {
            // compute the subsequent values using nExponent-1 to eventually reach the base case
            nextVal = Power(nBase, nExponent + 1);

            // multiply the base with all subsequent values
            returnVal = nBase * nextVal;
        }

        returnVal;
    }
}
}
```

Submission

Upload this completed document and the GitHub URL's of the 7 related project folders for #1, 2, 3, 4, 12, 13 and 14 to the corresponding myCourses dropbox.