**Unit Test #1**

**Due Wednesday 16-Sept-2020 11:59pm**

* 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:

* 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 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:

* Create a console application that uses a delegate to impersonate the Console.ReadLine() function when asking for user input. (Refer to "Math Delegate" or the attached "Number Sorter" application for example delegate code).

GitHub URL:

<https://github.com/jag8932/myIGME-201/tree/master/ReadLine_Delegate>

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

GitHub URL: <https://github.com/jag8932/myIGME-201/tree/master/Goodwillie_Midterm>

* Which of the following conversions can't be performed implicitly:
* ulong to int: Cannot be done implicitly.
* uint to long: Can be done.
* bool to string: Cannot be done implicitly.
* float to byte: Cannot be done implicitly.
* Given the following namespaces:

namespace Worst

{

using When = Episode;

// name "Meme" defined

namespace Episode

{

// name "Ever" defined

}

}

// global code

* How would you reference "Ever" from the global code? Worst.Episode.Ever
* How would you reference "Ever" from the "Worst" namespace using the alias?

When.Ever

* How would you reference "Ever" from the "Episode" namespace?

Ever

* How would you reference "Meme" from the global code?

Worst.Meme

* How would you reference "Meme" from the "Worst" namespace?

When.Meme

* How would you reference "Meme" from the "Episode" namespace?

Meme

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

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

resultVar will subtract the equation on the right.

1. var1 is added with var2.

2. The number they crreate is multiplied with var3.

3. That number is divided by var4 and its remainder(the mod of var3 and var4) then subtracts var5.

4. resultVar subtracts the resulting number.

* Given the formula z = 4y3 + 2x2 - 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.

double[,,] function = new double[40, 21, 3];

double dx, dy, dz;

int nX, nY;

nX = 0;

for (dx = 0; dx <= 4; dx += .1) {

dx = Math.Round(dx, 1);

nY = 0;

for(dy = -1; dy <= 1; dy += .1) {

dy = Math.Round(dy, 1);

dZ = 4 \*dy\*dy\*dy + 2\*dx\*dx - 8\*dx + 7;

dZ = Math.Round(dZ, 2);

function[nx, ny, 0] = dx;

function[nx, ny, 1] = dy;

function[nx, ny, 2] = dz;

nY++;

}

nX++;

}

* List all of the errors in the following code (Hint: there are 4 errors):

double[][] dArray = new int[2][]; //Cannot implicitly convert from double. Also missing the second array length declaration.

dArray[1] = new double[2];

dArray[2] = new double[1] //Missing semicolon. Also index is out of bounds since arrays start at 0. Should be dArray[1].

dArray[0][0] = 15; // Out of bounds index, the second array hasn't been set to a given length.

dArray[1][2] = 5.67; // Same issue here

* What will happen if the following code executes and why?

byte byteVal;

short shortVal = -556;

byteVal = (byte)shortVal;

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

It will display the number 212. This is because a byte can only contain numbers from 0 to 255. ShortVal is negative and its absolute value is bigger than the maximum byte size. When it converts to a byte, it cuts off data which is why it displays as 212.

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

$salary: Not legal

MAX\_INT\_#: Legal

99bottlesOfBeerOnTheWall: Not legal

#hashtag: Not legal

excitement!: Not legal

good&evil: Not legal

* 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/jag8932/myIGME-201/tree/master/testing2>

* 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/jag8932/myIGME-201/tree/master/testing2>

* 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 and rewriting them correctly, or adding any additional code required to prevent any run-time or logical errors. Ensure the code is fixed to work as documented.

GitHub URL: <https://github.com/jag8932/myIGME-201/tree/master/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 factorial 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.