**Tran Nguyen Quoc Khang**

**Instructions:**

* All questions should be answered in C#.
* Standard .NET libraries are allowed, but no third party libraries or SDKs, including Unity
* Deliver responses in plain text files identified with the question you’re answering
* Implement the methods using the given function signatures exactly
* Code will be evaluated for correctness, efficiency, time and memory complexity, and cleanness
* Code must function properly on all test cases for the problem, not just given examples
* Keep track of how long you spend on each problem and include it with your answer

**Question 1: Warm up**

Implement a function that plays the children’s game FizzBuzz: start counting from 1, saying each number in turn, but if the number is a multiple of 3 instead you say “Fizz”, if it’s a multiple of 5 you say “Buzz”, and if it’s a multiple of both you say “FizzBuzz”.

Your function should play the game by returning a single string with each count on a single line, in this format

1

2

Fizz

4

Buzz

Fizz

**Implement this function signature:**

**Answer: (15 ‘)**

public static string FizzBuzz(int turns)

{

StringBuilder output = new StringBuilder();

for(int i = 1; i <= n; i++)

{

if( i % 3 == 0 && i % 5 == 0)

{

output.AppendLine("FizzBuzz");

continue;

}

if( i % 3 == 0)

{

output.AppendLine("Fizz");

continue;

}

if( i % 5 == 0)

{

output.AppendLine("Buzz");

continue;

}

output.AppendLine(i.ToString());

}

return output.ToString();

}

**Question 2:**

Given a two-dimensional matrix of sequential integers contained in an array, return the data as a comma-separated string where the matrix has been “unrolled” by navigating in a clockwise spiral starting at the top left.

**Implement this function signature:**

public static string UnrollMatrix(int[] matrix, int rows, int columns)

For example, to unroll this matrix:

0 1 2

3 4 5

6 7 8

would be this function call:

UnrollMatrix(new int[] {0,1,2,3,4,5,6,7,8}, 3, 3)

which should output this string:

0,1,2,5,8,7,6,3,4

**Answer: (1h)**

public static string UnrollMatrix(int[] matrix, int rows, int columns)

{

int rowMaxIndex = rows - 1;

int colMaxIndex = columns - 1;

int elementsCount = rows \* columns;

int padding = 0;

int col = 0;

int row = 0;

StringBuilder result = new StringBuider();

for(int i = 0; i < elementsCount; i++)

{

result.AppendFormat( i < elementsCount - 1 ? "{0}," : "{0}" ,nums[row \* columns + col]);

// Left

if(col < colMaxIndex - padding && row == padding)

{

col++;

}

// Right

else if(col > padding && row == rowMaxIndex - padding)

{

col--;

}

// Down

else if(row < rowMaxIndex - padding && col == colMaxIndex - padding)

{

row++;

}

// Up

else if(row > padding + 1 && col == padding)

{

row--;

}

// End of a circle

else if(col == padding && row == padding + 1)

{

col++;

padding++;

}

}

return result.ToString();

}

**Question 3:**

Given an array of Points representing a mesh, optimize the mesh by collapsing any Points that are approximately equal within a given epsilon into a single Point at the average of those Points positions. Return an array of Points representing the optimized mesh.

There are multiple solutions that fulfill the requirements that may return slightly different answers given dense mesh data or a large epsilon, so please include a function comment explaining in a few sentences about how your algorithm is intended to solve the problem.

**Assume this struct exists:**

public struct Point

{

public float x;

public float y;

public float z;

public bool ApproxEquals(Point p, float epsilon = 0.00001f)

{

return (Math.Abs(x - p.x) < epsilon) && (Math.Abs(y - p.y) < epsilon) && (Math.Abs(z - p.z) < epsilon);

}

/\* SNIP assume all arithmetic operators are implemented \*/

}

**Implement this function signature:**

public static Point[] OptimizeMesh(Point[] mesh, float epsilon)

For example, given this set of points and an epsilon of 0.1:

2.07, 1.0, 1.0

2.13, 1.0, 1.0

1.0, 2.07, 1.0

1.0, 2.13, 1.0

3.01, 3.01, 3.01

2.99, 2.99, 2.99

3.0, 3.0, 3.0

the function should return this set of points:

2.1, 1.0, 1.0

1.0, 2.1, 1.0

3.0, 3.0, 3.0

**Answer: ( 2 h 30’ )**

public static Point[] OptimizeMesh(Point[] mesh, float epsilon)

{

    List<List<Point>> pointsGroup = new();

    for (int i = 0; i < mesh.Length; i++)

    {

        List<int> groupIndexes = new();

        Point currPoint = mesh[i];

        // The idea is to put points that are close together into lists.

        // For every point, traverse through those list,

        for (int groupIndex = 0; groupIndex < pointsGroup.Count; groupIndex++)

        {

            foreach (var point in pointsGroup[groupIndex])

            {

                if (currPoint.ApproxEquals(point, epsilon))

                {

                    groupIndexes.Add(groupIndex);

                    break;

                }

            }

        }

        // If current point belongs only to one list, add that point to the list.

        if (groupIndexes.Count == 1)

        {

            pointsGroup[groupIndexes[0]].Add(currPoint);

        }

        // If current point belongs to multiple lists, merge those lists together, then add the point the the merged list.

        // This is account for points that are not close to each other, but have the same point that are close to them.

        else if (groupIndexes.Count > 1)

        {

            List<Point> beginPoints = pointsGroup[groupIndexes[0]];

            for (int groupIndex = groupIndexes.Count - 1; groupIndex >= 1; groupIndex--)

            {

                int pointIndex = groupIndexes[groupIndex];

                List<Point> groupToAdd = pointsGroup[pointIndex];

                beginPoints.AddRange(groupToAdd);

                pointsGroup.RemoveAt(pointIndex);

            }

            beginPoints.Add(currPoint);

        }

        // Start it own list if it doesn't belong to any

        else

        {

            pointsGroup.Add(new List<Point> { currPoint });

        }

    }

    // Calculate average points

    List<Point> newMesh = new();

    Point averagePoint;

    foreach (var points in pointsGroup)

    {

        averagePoint.x = averagePoint.y = averagePoint.z = 0;

        foreach (var point in points)

        {

            averagePoint.x += point.x;

            averagePoint.y += point.y;

            averagePoint.z += point.z;

        }

        averagePoint.x /= points.Count;

        averagePoint.y /= points.Count;

        averagePoint.z /= points.Count;

        newMesh.Add(averagePoint);

    }

    return newMesh.ToArray();

}

**Question 4: Cooldown**

Implement a function that returns the nth number in the Fibonacci sequence starting at 0, where a given value in the sequence is the sum of the previous two values (0, 1, 1, 2, 3, 5, 8, 13, etc)

**Implement this function signature:**

**Answer: (20’)**

public static int Fib(int n)

{

if( n <= 1 )

return n;

int[] fibArray = new int[n + 1];

fibArray[0] = 0;

fibArray[1] = 1;

for(int i = 2; i <= n; i++)

{

fibArray[i] = fibArray[i - 1] + fibArray[i - 2];

}

return fibArray[n];

}