For the C# Challenge, you'll have to solve the following problem:

You have multiple shapes and any shape has the following 2 methods:

- void DrawShape(), which prints a string to standard output
- double GetSurface(), which returns the surface area of the shape

The shapes are:

1. Rectangle
    a. Has 2 fields: height and width
    b. Constructor sets both fields
    c. DrawShapes prints: Rectangle: <height> x <width>
    d. GetSurface()
2. Square
    a. Has 1 field: length
    b. Constructor sets field
    c. DrawShapes prints: Square: <length> x <length>
    d. GetSurface()
3. Triangle
    a. Has 3 fields: sideA, sideB and sideC
    b. Constructor sets all fields
    c. DrawShapes prints: Triangle: A = <a>, B = <b>, C = <c>
    d. GetSurface()
4. Circle
    a. Has 1 field: radius
    b. Constructor sets field
    c. DrawShapes prints: Circle: R = <radius>
    d. GetSurface()
5. (Bonus) Add any shape you want and implement the appropriate methods

No fields should be accessible from outside of the class. For Circle, you can either choose pi as 3.14 or use a math library. Implement as many other methods as you consider useful.

There is a class called ShapeRepository which contains a private list of shapes of any type and a "limit" field. The constructor sets the limit, which you will use in the AddShape() method and should not be able to be modified later. ShapeRepository implements the following methods:

1. bool AddShape(Shape shape): adds shape with a surface area greater than the limit to the list and returns true if the shape was added or false if it wasn't
2. bool RemoveShape(Shape shape): removes a shape from the list and returns true if the shape was removed or false if it wasn't found
3. void DrawAllShapes(): draws each shape from the list
4. double GetTotalSurface(): returns the sum of all shapes' surfaces or 0 if there aren't any shapes
5. double GetMaxSurface(): returns the largest surface of any shape in the list or 0
6. List<Shape> GetAllShapesSmallerThan(double upperLimit): returns a list containing all shapes with a smaller surface than the required limit (!!bonus points if you figure out the hidden condition!!)
7. Dictionary<string, List<Shapes>> MapAllShapesToType(): returns a dictionary having the key a type of shape (circle, triangle, etc.) and the value a list of all shapes of that type
   *try to limit the number of iterations used

You are free to implement other functionalities than these. Try to implement as many of the required methods as possible. You will be awarded bonus points for using concepts which were not explained during the workshop: (interfaces, exceptions, lambda functions, readonly, etc.).

You will upload all files in an archive with this format: **DotNet_FirstName_LastName.zip/tar.gz/7z**. Send them on WhatsApp or email to Veniamin Balan - +37379900846 / veniamin.balan02@gmail.com.

Good luck and have fun!