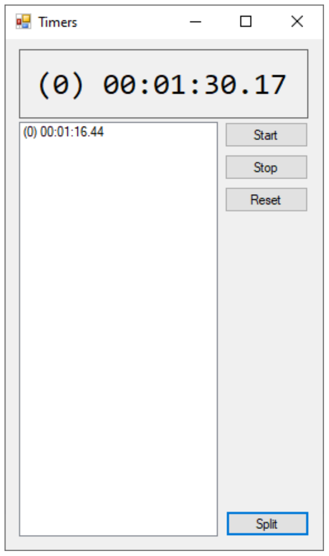
School of Applied Sciences And Technology
Department of IST
Program: CNT
CMPE1666 – ICA 03-Kind-of-sort-of-Timing

In this ICA you will explore two mechanisms of measuring time, namely **timers** and the **stopwatches**.

Create an application with the following appearance:



- The top control is a label, non-autosized, with a border. Note: the font is non-proportional, in this case, Consolas.
- The bottom-left control is a list box, with default settings, except the 'integral height' property must be switched to **false** to make it easier for layout.
- The buttons are just basic buttons.
- The anchoring permits width gain for the label and list box, but height gain for the list box only. Buttons stick to the form edges on resize, but don't resize.
- The time display is up to hundredth of milliseconds. The value in () is the number of split that has occurred.

The application will manage all time measurement with a `System.Diagnostics.Stopwatch` that you will create at class-level in the form.

The top three buttons will operate the stated stopwatch operation (use dot exploration or traditional research to learn about the `Stopwatch` class).

You will use a `Timer` component to periodically update the UI. The timer will in no way be used to measure time – its sole purpose is to maintain the UI. Use a timer tick rate of no more than 20ms. Humans don't need more than 50 updates per second (although gamers may start a fistfight over this).

In your timer tick event, show the current elapsed time from the stopwatch. The stopwatch may report elapsed time in milliseconds, so this is a good, high-enough resolution for us to display hundredths of a second.

You will be showing the stopwatch time in two locations in your code, so it makes sense to create a method to produce the formatted string that is the elapsed time that you see above. This method should take an argument of milliseconds and return a formatted string. Do **not** use the **Elapsed** property of the **Stopwatch** class here. We want you to use this opportunity to demonstrate that you can create a helper function using a little math. You will still need string interpolation to format the string and will still need to do some math to correctly create the hundredths of a seconds output. Incremental testing here will be key to avoiding lost time debugging later.

When the ‘Split’ button is clicked, you will add the current, formatted elapsed time to the list box. You will not permit duplicates in the list box.

The ‘Reset’ button will set the time to 0 and clear the contents of the list box.

The Tab Stop order must be: start->split->stop->Reset

Rubric [30 Marks]

Item	Marks	Penalties
UI Design (33%) • UI is as directed. • Tab Order is Correct. • Anchoring is as discussed. • Format for time is correct.	10	Inappropriate control names: -3 Top control allowing autosize: -2 Anchoring not as required: -4 Wrong time formatting: -3
Code Design and Implementation (67%) • Time formatting code is in a separate method, takes a number of milliseconds and performs conversion • Time displayed in label • Time displayed in listbox on split • Start and stop buttons function as required • Reset button function as required • Duplicates not permitted in list box.	5 3 3 3 3 3	
Documentation: • Programmer Block • Well commented code • Appropriate Variable Names • Proper spacing between blocks of code • Control names are consistent and appropriate.		Penalties: -1 to -6 based on instructor’s judgement