```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace ExpoCalc
{
    public partial class Form1 : Form
    {
        private Int32 inputRangeLow;
        private Int32 inputRangeHigh;
        private Int32 outputRangeLow;
        private Int32 outputRangeHigh;
        private Int32 expo;

        private double expoAdjustedRangeLower = 0.0f;
        private double expoAdjustedRangeHigher = 0.0f;

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }

        private void buttonCalculate_Click(object sender, EventArgs e)
        {
            if (ValidateInputs())
            {
                CalculateExpoValues();
            }
        }

        private bool ValidateInputs()
        {
            try
            {
                inputRangeLow = (Int32)nudInputLower.Value;
                inputRangeHigh = (Int32)nudInputHigher.Value;
                outputRangeLow = (Int32)nudOutputLower.Value;
                outputRangeHigh = (Int32)nudOutputHigher.Value;
                expo = (Int32)nudExpo.Value;

                if (inputRangeLow > inputRangeHigh)
                    throw new ArgumentOutOfRangeException("At the moment, Input Range Low must be smaller    ↙
    than Input Range High");
                if (outputRangeLow > outputRangeHigh)
                    throw new ArgumentOutOfRangeException("At the moment, Output Range Low must be smaller   ↙
    than Ouput Range High");
            }
            catch(Exception ex)
            {
                MessageBox.Show("Please check inputs:" + ex.Message);
                return false;
            }
            return true;
        }

        private void CalculateExpoValues()
        {
            lbOutput.Items.Clear();
            /** get the lower and higher values of the ranges
             *  When we apply expo the upper and lower ranges change,
             *  based on the expo setting the upper range may go up to the value (Natural Antilog) e (limit  ↙
    infinity)
             *  **/
            expoAdjustedRangeLower = GetExpoAdjustedValue(inputRangeLow);
            expoAdjustedRangeHigher = GetExpoAdjustedValue(inputRangeHigh);

            if (inputRangeLow < inputRangeHigh)
            {
                Int32 lower = inputRangeLow - 1;
                /** go in the positive direction **/
                while (++lower <= inputRangeHigh)
                {

                    double transformedValue  = 0.0f;
                    if (expo == 0)
                    {
                        transformedValue = outputRangeLow + (((double)lower- inputRangeLow) / (inputRangeHigh↙
    - inputRangeLow)) * (outputRangeHigh - outputRangeLow); ; //prevent divide by 0;
                    }
                    else
                    {
                        double expoAdjustedValue = GetExpoAdjustedValue(lower);
                        transformedValue = outputRangeLow + ((expoAdjustedValue - expoAdjustedRangeLower) /  ↙
    (expoAdjustedRangeHigher - expoAdjustedRangeLower)) * (outputRangeHigh - outputRangeLow);
                    }

                    /** we now have the updated exp ranges and a value in that range,
                     * apply a lerp to get interpoalted value on the target range
                     * **/


                    lbOutput.Items.Add(String.Format("Input: {0:F4} -> Output {1:F4}", lower,              ↙
    transformedValue));
                }
            }


        }
        private double GetExpoAdjustedValue(double value)
        {
            double valueMagnitude = value - inputRangeLow;
            double rangeMagnitude = inputRangeHigh - inputRangeLow;

            double ratio = valueMagnitude / rangeMagnitude;
            return Math.Pow((double)(1 + ratio / expo), (double)expo);
        }
    }
}
```