College LaSalle

Project - Oriented Object Programming User and Technical Manual

Presented to: Mihai Maftei.

Your name: Ayrton Senna Seraphim do Amaral

4/17/2023
Version: 6.0

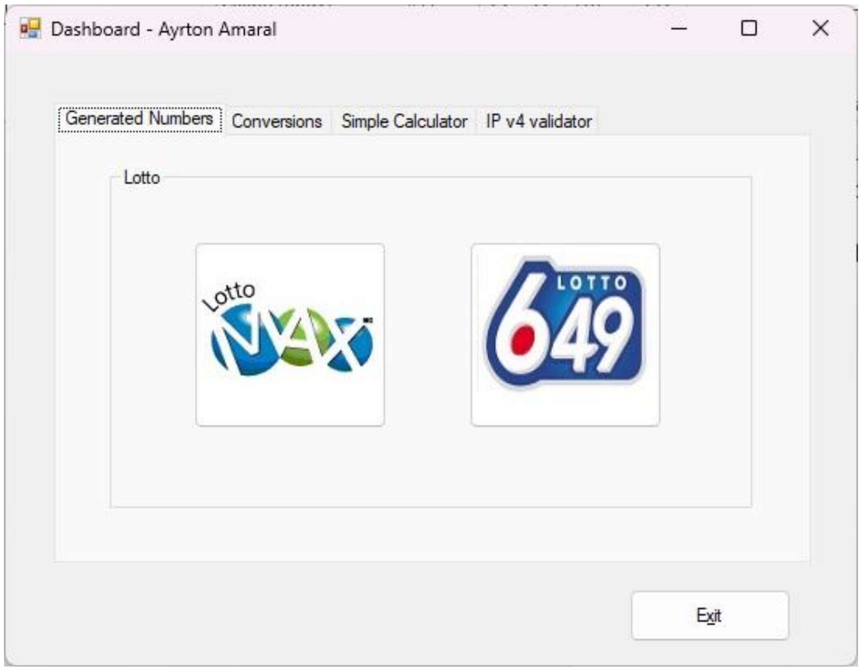
A. Start by adding a short description of your project, and the languages (technologies) used:

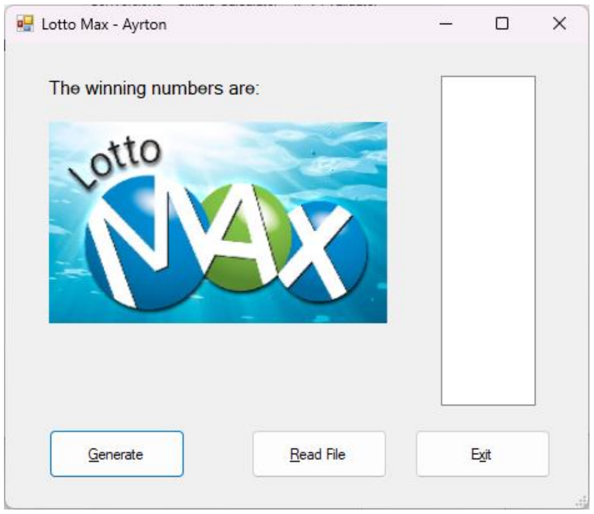
1. Language: C#
2. Tools (IDE): Visual Studio 2022

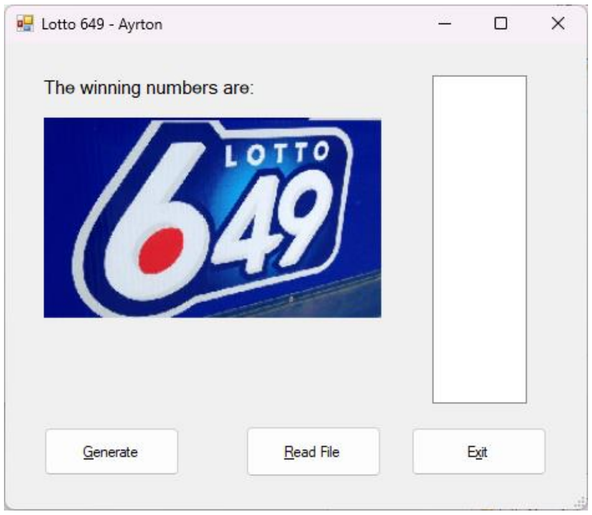
Multiform (.NET Framework) project, covers 6 application.

B. Present the print screens of yours forms, and have a detailed description of the functionalities (step by step).

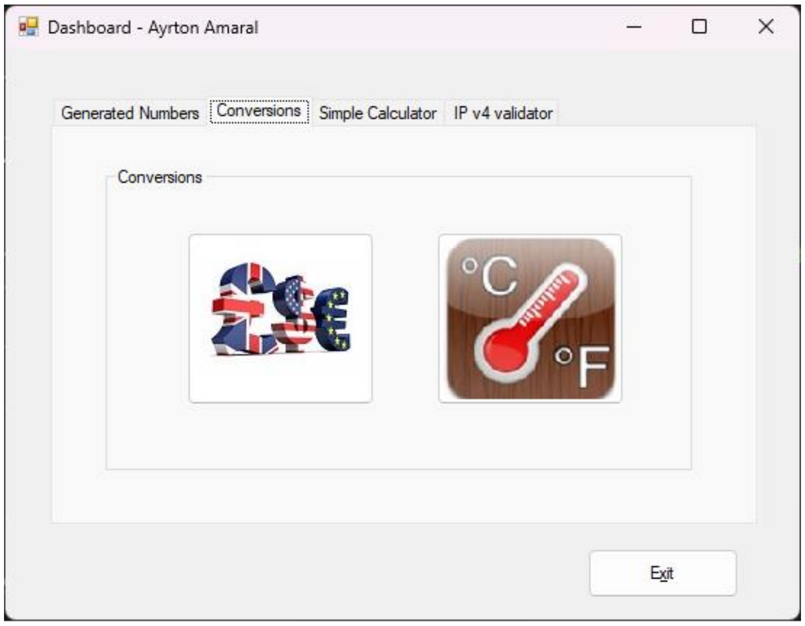
- In the Dashboard it is possible to choose which of the 6 applications the user wants to use.

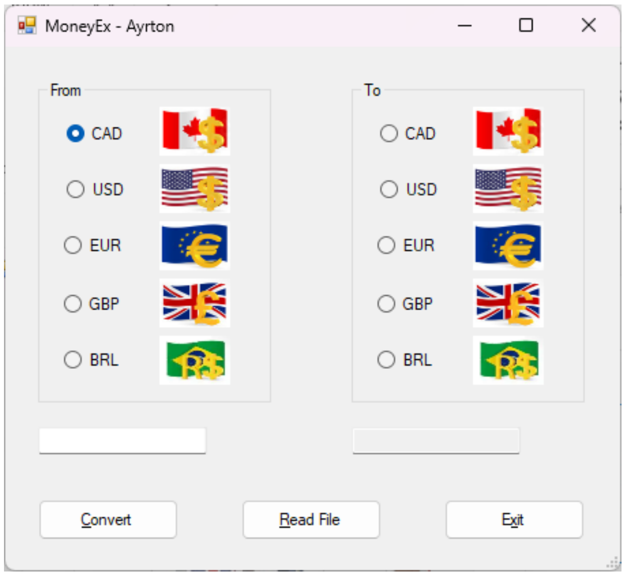




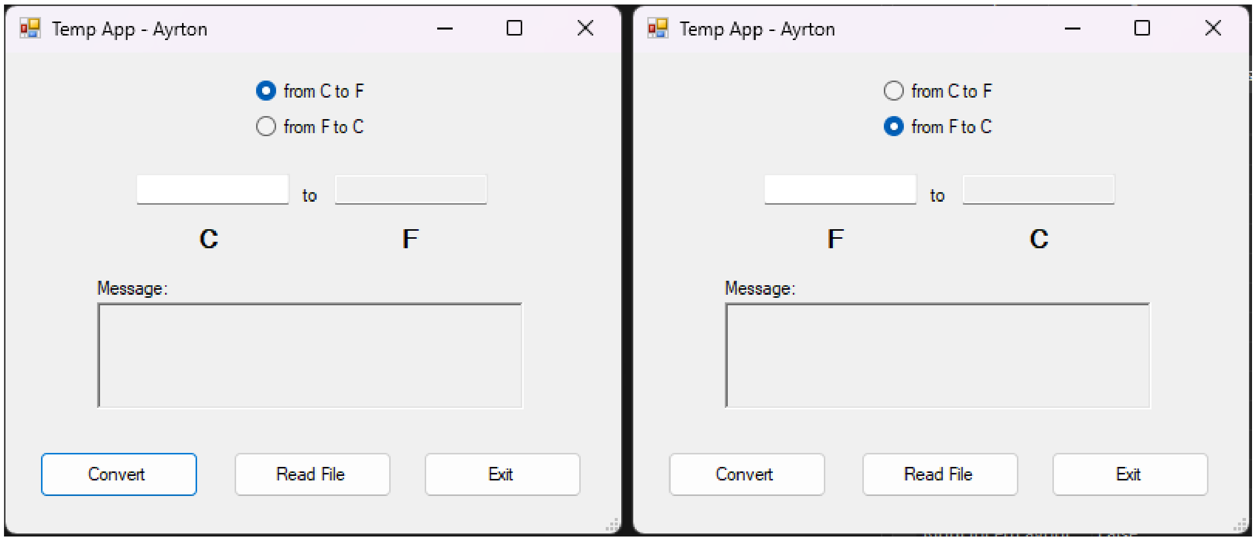


1. If you click on the button Generate in LottoMax/Lotto649 it will display the random numbers in the listbox located on the right.
2. The event Generate is also writing in a textfile (both LottoMax and Lotto649) the registers of each generation.
3. If you click on the button Read File, it will open a MessageBox with the previous registers per line of each Generate event realized before.
4. If you click on the Exit button it will open a MessageBox checking if you really want to perform this action. If you accept it closes the application and returns to the main one.



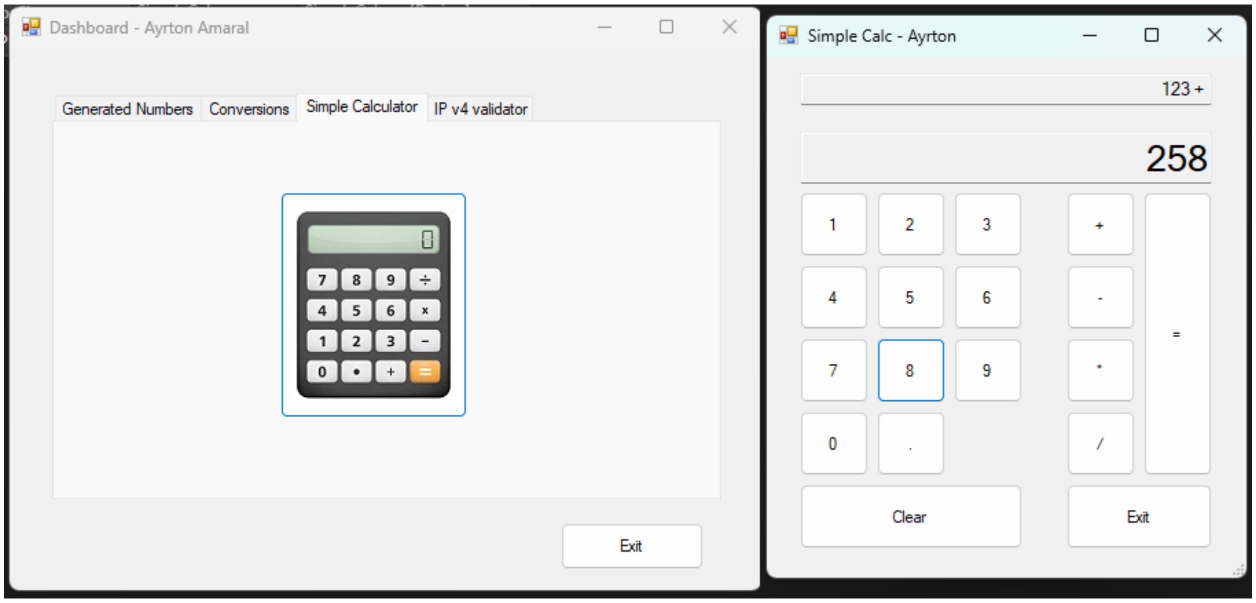


5. In this application you choose a radiobutton on the left to perform a conversion of money. The ‘From’ radiobuttons are the currency you have and the ‘To’ radiobuttons are the one you want to exchange.
6. The action will be performed after selecting the desired radiobuttons (of both columns) and after inserting the value you want to convert inside the textbox located on the left.
7. Then, you press convert and the converted value will appear inside the read only textbox, located on the right.
8. The Convert event also registers all the conversions operated in a text file.
9. When you change the checked radiobuttons after a conversion it will clean the textbox that contains the converted value.
10. The Read File button will open the register of all the previous conversions made.
11. If you click on the Exit button it will open a MessageBox checking if you really want to perform this action. If you accept it closes the application and returns to the main one.

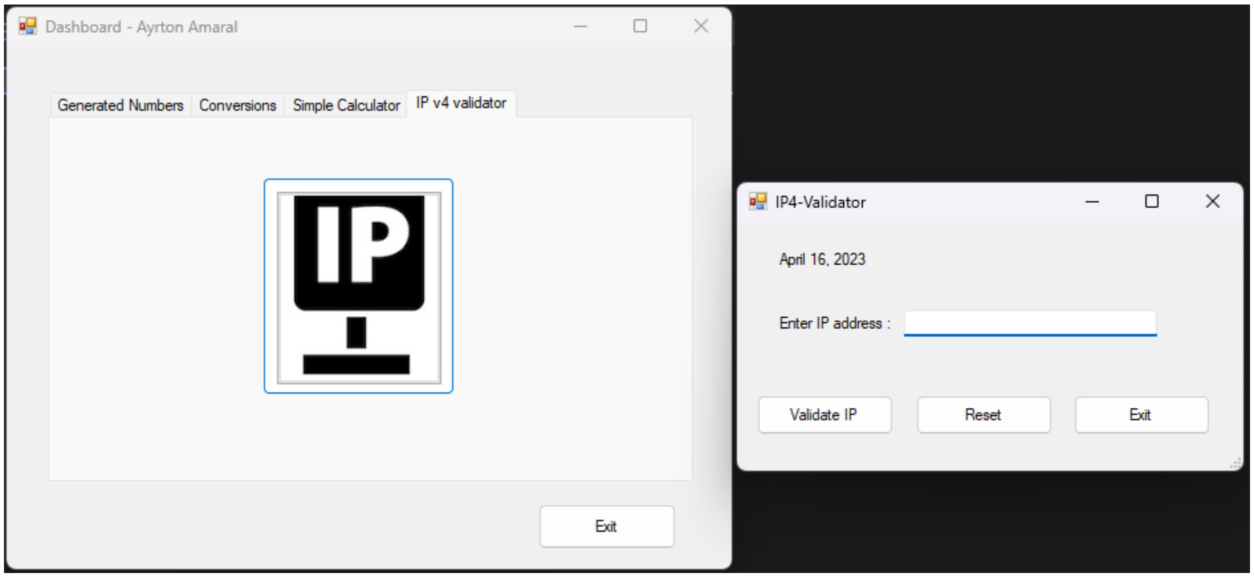


12. In the Temp App - Ayrton you choose a radiobutton (‘from C to F’ or ‘from F to C’) to perform a temperature conversion.
13. Depending on the radiobutton you select you may see the labels below the textboxes changing their order between C F and F C.
14. The action will be performed after selecting the desired radiobutton and after inserting the value (temperature) to convert inside the textbox located on the left.
15. Then, you press convert and the converted value will appear inside the read only textbox, located on the right.
16. Depending on the result you may see in the field called ‘Message’ specific message description about the temperature obtained.
17. The Convert event also registers all the conversions operated in a text file.

18. When you change the checked radiobuttons after a conversion it will clean the textbox that contains the converted value.
19. The Read File button will open the register of all the previous conversions made.
20. If you click on the Exit button it will open a MessageBox checking if you really want to perform this action. If you accept it closes the application and returns to the main one.



21. The Simple Calculator application performs events for each operator seen +-*/
22. There is a textbox that stores during the operation what is being calculated at the moment with =
23. The numbers were unified in an event, so when you click the numbers it is going to appear on the screen, no matter how many times you to click on it.
24. There is a 'Clear' button to clean the calculation realized or in progress that, as the name says, clean the numbers (appearing on the bigger textbox) pressed before.
25. It is not possible to see, but there is also a reading into text files happening in this application. The simple calculations performed are being stored in a .txt file.



26. In the IP4-Validator you enter the IP you want to validate in the v4 format inside the only textbox available.
27. When you press ‘Validate IP’ it will perform the event and then display a MessageBox saying if it is valid or not. If invalid, you may try to correct your IP address.
28. In this application you will find a ‘Reset’ button that clears the textbox if there’s a previous text inside.
29. If you click on the Exit button it will open a MessageBox checking if you really want to perform this action. If you accept it closes the application and returns to the main one.
30. One different aspect of this application is that it displays your actual datetime when you load the form and when you confirm to exit it is going to show the time you spent using the application in minutes and seconds.

C. Present the code of your application (forms).

… your code goes here.

```
LOTTOMAX:
public partial class LottoMax : Form
    {
        public LottoMax()
        {
            InitializeComponent();
        }

        string path = @".\LottoNbrs.txt";

        private void btnGenerateMax_Click(object sender, EventArgs e)
        {
            listBoxLottoMax.Items.Clear();

            Random random = new Random();

            // HashSet is a type of collection that doesn't allow to repeat numbers
            HashSet<int> numbersGenerated = new HashSet<int>();

            // Genrating random numbers:
            do
            {
                numbersGenerated.Add(random.Next(1, 50));
            } while (numbersGenerated.Count < 8);

            // Inserting each element generated in the listBox:
            foreach (int element in numbersGenerated)
            {
                listBoxLottoMax.Items.Add(element);
            }

            // To organize the rows of the Text File:
            int[] arrNumbers = numbersGenerated.ToArray(); // To access each element 
as index

            DateTime currentDateTime = DateTime.Now;
            // Max, 2023/3/20 2:17:36 PM, 31,36,3,39,24,42,4 Bonus 30
            string textRows = $"Max, {currentDateTime}, {arrNumbers[0]}, 
{arrNumbers[1]}, {arrNumbers[2]}, " +
                                    $"{arrNumbers[3]}, {arrNumbers[4]}, 
{arrNumbers[5]}, {arrNumbers[6]}, Bonus {arrNumbers[7]}";

            // Write in Text File:
            FileStream fs = null;

            try
            {
                fs = new FileStream(path, FileMode.Append, FileAccess.Write);

                // Create the output stream for a text file that exists
                StreamWriter textOut = new StreamWriter(fs);

                // Write the following fields into text file:
                textOut.WriteLine(textRows);
                textOut.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
            finally
            {
                if (fs != null) fs.Close();
            }
        }

        private void btnReadMax_Click(object sender, EventArgs e)
        {
            FileStream fs = null;
            fs = new FileStream(path, FileMode.OpenOrCreate, FileAccess.Read);

            string textToDisplay = "";

            // Create the object for the input stream for a text file
            StreamReader streamReader = new StreamReader(fs);

            // Read the data from the file:
            while (streamReader.Peek() != -1)
            {
                string row = streamReader.ReadLine();

                textToDisplay += row + "\n";
            }

            streamReader.Close();
            fs.Close();

            MessageBox.Show(textToDisplay, "LottoMax - Ayrton");
        }

        private void btnExitLottoMax_Click(object sender, EventArgs e)
        {
            if (MessageBox.Show("Do you want to quit this application? ", "Exit ?", 
MessageBoxButtons.YesNo).ToString() == "Yes")
            {
                this.Close();
            }
        }
    }
```

```
LOTTO649:
public partial class Lotto649 : Form
    {
        public Lotto649()
        {
            InitializeComponent();
        }

        string path = @".\LottoNbrs.txt";

        private void btnGenerate649_Click(object sender, EventArgs e)
        {
            listBoxLotto649.Items.Clear();

            Random random = new Random();

            // HashSet is a type of collection that doesn't allow to repeat numbers
            HashSet<int> numbersGenerated = new HashSet<int>();

            // Genrating random numbers:
            do
            {
                numbersGenerated.Add(random.Next(1, 49));
            } while (numbersGenerated.Count < 7);

            // Inserting each element generated in the listBox:
            foreach (int element in numbersGenerated)
            {
                listBoxLotto649.Items.Add(element);
            }

            // To organize the rows of the Text File:
            int[] arrNumbers = numbersGenerated.ToArray(); // To access each element
as index

            DateTime currentDateTime = DateTime.Now;

            // 649, 2023/3/20 2:17:42 PM, 29,45,42,22,41,18 Bonus 11
            string textRows = $"649, {currentDateTime}, {arrNumbers[0]},
{arrNumbers[1]}, {arrNumbers[2]}, " +
                                            $"{arrNumbers[3]}, {arrNumbers[4]},
{arrNumbers[5]}, Bonus {arrNumbers[6]}";

            // Write in Text File:
            FileStream fs = null;

            try
            {
                fs = new FileStream(path, FileMode.Append, FileAccess.Write);

                // Create the output stream for a text file that exists
                StreamWriter textOut = new StreamWriter(fs);

                // Write the following fields into text file:
                textOut.WriteLine(textRows);
                textOut.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
            finally
            {
                if (fs != null) fs.Close();
            }
        }

        private void btnRead649_Click(object sender, EventArgs e)
        {
            FileStream fs = null;
            fs = new FileStream(path, FileMode.OpenOrCreate, FileAccess.Read);

            string textToDisplay = "";

            // Create the object for the input stream for a text file
            StreamReader streamReader = new StreamReader(fs);

            // Read the data from the file:
            while (streamReader.Peek() != -1)
            {
                string row = streamReader.ReadLine();

                textToDisplay += row + "\n";
            }

            streamReader.Close();
            fs.Close();

            MessageBox.Show(textToDisplay, "Lotto649 - Ayrton");
        }

        private void btnExitLotto649_Click(object sender, EventArgs e)
        {
            if (MessageBox.Show("Do you want to quit this application? ", "Exit ?",
MessageBoxButtons.YesNo).ToString() == "Yes")
            {
                this.Close();
            }
        }
    }
```

MONEY EXCHANGE:

```
internal class Currency
    {
        public double CanadianFactor { get; }

        public string CurrencyName { get; }

        public Currency(double canadianFactor, string currencyName)
        {
            CanadianFactor = canadianFactor;
            CurrencyName = currencyName;
        }

        public double ConvertToCAD(double valueInserted)
        {
            return valueInserted * CanadianFactor;
        }

        public double FromCADtoCurrency(double cadValue)
        {
            return cadValue / CanadianFactor;
        }
    }
```

```
public partial class MoneyExchange : Form
    {
        public MoneyExchange()
        {
            InitializeComponent();
        }

        string pathMoneyEx = @".\MoneyConversions.txt";

        private void MoneyExchange_Load(object sender, EventArgs e)
        {
            radioFromCAD.Checked = true;
            radioToCAD.Checked = true;

            textBox1.Text = "0";
        }

        private void btnConvertMoney_Click(object sender, EventArgs e)
        {
            // Instanciating objects of currency:
            Currency cad = new Currency(1, "CAD");
            Currency usd = new Currency(1.36, "USD");
            Currency eur = new Currency(1.50, "EUR");
            Currency gbp = new Currency(1.65, "GBP");
            Currency brl = new Currency(0.28, "BRL");

            double valueInserted = Convert.ToDouble(textBox1.Text);
            double valueIncad = 0;
            double convertedValue = 0;

            string currencyDisplayFrom = "";
            string currencyDisplayTo = "";
```

```
// From:
if (radioFromCAD.Checked)
{
    valueIncad = cad.ConvertToCAD(valueInserted);
    currencyDisplayFrom = cad.CurrencyName;
}
if (radioFromUSD.Checked)
{
    valueIncad = usd.ConvertToCAD(valueInserted);
    currencyDisplayFrom = usd.CurrencyName;
}
if (radioFromEUR.Checked)
{
    valueIncad = eur.ConvertToCAD(valueInserted);
    currencyDisplayFrom = eur.CurrencyName;
}
if (radioFromGBP.Checked)
{
    valueIncad = gbp.ConvertToCAD(valueInserted);
    currencyDisplayFrom = gbp.CurrencyName;
}
if (radioFromBRL.Checked)
{
    valueIncad = brl.ConvertToCAD(valueInserted);
    currencyDisplayFrom = brl.CurrencyName;
}

// To:
if (radioToCAD.Checked)
{
    convertedValue = cad.FromCADtoCurrency(valueIncad);
    currencyDisplayTo = cad.CurrencyName;
}
if (radioToUSD.Checked)
{
    convertedValue = usd.FromCADtoCurrency(valueIncad);
    currencyDisplayTo = usd.CurrencyName;
}
if (radioToEUR.Checked)
{
    convertedValue = eur.FromCADtoCurrency(valueIncad);
    currencyDisplayTo = eur.CurrencyName;
}
if (radioToGBP.Checked)
{
    convertedValue = gbp.FromCADtoCurrency(valueIncad);
    currencyDisplayTo = gbp.CurrencyName;
}
if (radioToBRL.Checked)
{
    convertedValue = brl.FromCADtoCurrency(valueIncad);
    currencyDisplayTo = brl.CurrencyName;
}

textBox2.Text = Math.Round(convertedValue, 2).ToString();
```

```
            // Organize the text file:
            DateTime currentDateTime = DateTime.Now;

            // 100 EUR = 85 USD, 2023 / 3 / 29 10:07:03 AM
            string textRows = $"{valueInserted} {currencyDisplayFrom} =
{textBox2.Text} {currencyDisplayTo}, {currentDateTime}";


            // Write in Text File:
            FileStream fs = null;

            try
            {
                fs = new FileStream(pathMoneyEx, FileMode.Append, FileAccess.Write);

                // Create the output stream for a text file that exists
                StreamWriter textOut = new StreamWriter(fs);

                // Write the following fields into text file:
                textOut.WriteLine(textRows);
                textOut.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
            finally
            {
                if (fs != null) fs.Close();
            }
        }

        private void btnReadMoneyEx_Click(object sender, EventArgs e)
        {
            FileStream fs = null;
            fs = new FileStream(pathMoneyEx, FileMode.OpenOrCreate,
FileAccess.Read);

            string textToDisplay = "";

            // Create the object for the input stream for a text file
            StreamReader streamReader = new StreamReader(fs);

            // Read the data from the file:
            while (streamReader.Peek() != -1)
            {
                string row = streamReader.ReadLine();

                textToDisplay += row + "\n";
            }

            streamReader.Close();
            fs.Close();

            MessageBox.Show(textToDisplay, "MoneyEx - Ayrton");
        }

        private void btnExitMoneyEx_Click(object sender, EventArgs e)
        {
            if (MessageBox.Show("Do you want\nto quit the application\nMoney
Exchange?", "Exit ?", MessageBoxButtons.YesNo).ToString() == "Yes")
            {
                this.Close();
            }
        }

        private void radioFrom_CheckedChanged(object sender, EventArgs e)
        {
            this.textBox2.Text = "";
        }

        private void radioTo_CheckedChanged(object sender, EventArgs e)
        {
            this.textBox2.Text = "";
        }
    }
```

TEMPERATURE CONVERTER:

```
public partial class TemperatureConverter : Form
    {
        public TemperatureConverter()
        {
            InitializeComponent();
        }

        string path = @".\TempConversions.txt";

        private void radioButton1_CheckedChanged(object sender, EventArgs e)
        {
            if (radioButton1.Checked)
            {
                label3.Text = "C";
                label4.Text = "F";
                textBox2.Clear();
            }
        }

        private void radioButton2_CheckedChanged(object sender, EventArgs e)
        {
            if (radioButton2.Checked)
            {
                label3.Text = "F";
                label4.Text = "C";
                textBox2.Clear();
            }
        }

        private void btnConvertTemp_Click(object sender, EventArgs e)
        {
            double tempToConvert, convertedTemp;
            string textRows = "";
            DateTime currentDateTime = DateTime.Now;

            Regex myRegex = new Regex(@"^-?\d+(?:\.\d)?$");
            if (myRegex.IsMatch(textBox1.Text.Trim()) == true)
            {
                tempToConvert = Convert.ToDouble(textBox1.Text.Trim());

                if (radioButton1.Checked)
                {
                    // Celsius to Fahrenheit:
                    // (°C × 9/5) + 32 = °F
                    convertedTemp = (tempToConvert * 9 / 5) + 32;
                    convertedTemp = Math.Round(convertedTemp, 1);
                    textBox2.Text = convertedTemp.ToString();

                    if(convertedTemp == 212)
                    {
                        richTextBox.Text = "Water boils";
                        richTextBox.ForeColor = Color.DarkRed;
                    }
                    else if (convertedTemp == 104)
                    {
                        richTextBox.Text = "Hot Bath";
                        richTextBox.ForeColor = Color.Red;
                    }
```

```
                        else if (convertedTemp == 98.6)
                        {
                            richTextBox.Text = "Body temperature";
                            richTextBox.ForeColor = Color.DarkOrange;
                        }
                        else if (convertedTemp == 86)
                        {
                            richTextBox.Text = "Beach weather";
                            richTextBox.ForeColor = Color.Orange;
                        }
                        else if (convertedTemp > 50 && convertedTemp < 86)
                        {
                            richTextBox.Text = "Room temperature";
                            richTextBox.ForeColor = Color.Green;
                        }
                        else if (convertedTemp == 50)
                        {
                            richTextBox.Text = "Cool Day";
                            richTextBox.ForeColor = Color.DodgerBlue;
                        }
                        else if (convertedTemp == 32)
                        {
                            richTextBox.Text = "Freezing point of water";
                            richTextBox.ForeColor = Color.DarkBlue;
                        }
                        else if (convertedTemp > -40 && convertedTemp < 32)
                        {
                            richTextBox.Text = "Very Cold Day";
                            richTextBox.ForeColor = Color.DarkBlue;
                        }
                        else if (convertedTemp == -40)
                        {
                            richTextBox.Text = "Extremely Cold Day\n(and the same 
number!)";

                            richTextBox.ForeColor = Color.Purple;
                        }

                        textRows = $"{tempToConvert} C = {convertedTemp} 
F,\t{currentDateTime}\t{richTextBox.Text.Replace("\n", " ")}";
                    }

                    if (radioButton2.Checked)
                    {
                        // Fahrenheit to Celsius:
                        // (°F − 32) x 5/9 = °C
                        convertedTemp = (tempToConvert - 32) * 5 / 9;
                        convertedTemp = Math.Round(convertedTemp, 1);
                        textBox2.Text = convertedTemp.ToString();

                        if (convertedTemp == 100)
                        {
                            richTextBox.Text = "Water boils";
                            richTextBox.ForeColor = Color.DarkRed;
                        }
                        else if (convertedTemp == 40)
                        {
                            richTextBox.Text = "Hot Bath";
                            richTextBox.ForeColor = Color.Red;
                        }
                        else if (convertedTemp == 37)
                        {
                            richTextBox.Text = "Body temperature";
                            richTextBox.ForeColor = Color.DarkOrange;
                        }
                        else if (convertedTemp == 30)
                        {
                            richTextBox.Text = "Beach weather";
                            richTextBox.ForeColor = Color.Orange;
                        }
                        else if (convertedTemp < 30 && convertedTemp > 10)
                        {
                            richTextBox.Text = "Room temperature";
                            richTextBox.ForeColor = Color.Green;
                        }
                        else if (convertedTemp == 10)
                        {
                            richTextBox.Text = "Cool Day";
                            richTextBox.ForeColor = Color.DodgerBlue;
                        }
                        else if (convertedTemp == 0)
                        {
                            richTextBox.Text = "Freezing point of water";
                            richTextBox.ForeColor = Color.DarkBlue;
                        }
                        else if (convertedTemp > -40 && convertedTemp < 0)
                        {
                            richTextBox.Text = "Very Cold Day";
                            richTextBox.ForeColor = Color.DarkBlue;
                        }
                        else if (convertedTemp == -40)
                        {
                            richTextBox.Text = "Extremely Cold Day\n(and the same
    number!)";

                            richTextBox.ForeColor = Color.Purple;
                        }

                        textRows = $"{tempToConvert} F = {convertedTemp}
    C,\t{currentDateTime}\t{richTextBox.Text.Replace("\n", " ")}";
                    }
                }

                // Write in Text File:
                FileStream fs = null;

                try
                {
                    fs = new FileStream(path, FileMode.Append, FileAccess.Write);

                    // Create the output stream for a text file that exists
                    StreamWriter textOut = new StreamWriter(fs);

                    // Write the following fields into text file:
                    textOut.WriteLine(textRows);
                    textOut.Close();
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                }
                finally
                {
                    if (fs != null) fs.Close();
                }
            }

            private void btnReadTemp_Click(object sender, EventArgs e)
            {
                FileStream fs = null;
                fs = new FileStream(path, FileMode.OpenOrCreate, FileAccess.Read);

                string textToDisplay = "";

                // Create the object for the input stream for a text file
                StreamReader streamReader = new StreamReader(fs);

                // Read the data from the file:
                while (streamReader.Peek() != -1)
                {
                    string row = streamReader.ReadLine();

                    textToDisplay += row + "\n";
                }

                streamReader.Close();
                fs.Close();

                MessageBox.Show(textToDisplay, "TempApp - Ayrton");
            }

            private void btnExitTemp_Click(object sender, EventArgs e)
            {
                if (MessageBox.Show("Do you want to quit this application? ", "Exit ?", 
    MessageBoxButtons.YesNo).ToString() == "Yes")
                {
                    this.Close();
                }
            }
        }
```

CALCULATOR:

```
internal class CalculatorClass
    {
        // Private Fields:
        private decimal currentValue, operand1, operand2;
        private string op;

        // Public Properties:
        public decimal Operand1
        {
            get { return operand1; }
            set { operand1 = value; }
        }

        public decimal Operand2
        {
            get { return operand2; }
            set { operand2 = value; }
        }

        public decimal CurrentValue
        {
            get { return currentValue; }
            set { currentValue = value; }
        }

        // Default Constructor:
        public CalculatorClass()
        {
            op = null;
        }

        // Methods:
        public void Add(decimal displayValue)
        {
            CurrentValue = displayValue;
            operand1 = CurrentValue;
            op = "+";
        }

        public void Subtract(decimal displayValue)
        {
            CurrentValue = displayValue;
            operand1 = CurrentValue;
            op = "-";
        }

        public void Multiply(decimal displayValue)
        {
            CurrentValue = displayValue;
            operand1 = CurrentValue;
            op = "*";
        }

        public void Divide(decimal displayValue)
        {
            CurrentValue = displayValue;
            operand1 = CurrentValue;
            op = "/";
        }

        public void EqualsOp(decimal displayValue)
        {
            Operand2 = displayValue;

            switch (op)
            {
                case "+":
                    CurrentValue = Operand1 + Operand2;
                break;

                case "-":
                    CurrentValue = Operand1 - Operand2;
                break;

                case "*":
                    CurrentValue = Operand1 * Operand2;
                break;

                case "/":
                    CurrentValue = Operand1 / Operand2;
                break;

                default:
                break;
            }
        }

        public void Clear()
        {
            Operand1 = 0;
            Operand2 = 0;
            CurrentValue = 0;
            op = null;
        }
    }
```

```
public partial class SimpleCalc : Form
    {
        public SimpleCalc()
        {
            InitializeComponent();
        }

        private CalculatorClass calculator = new CalculatorClass();

        private void buttonNbrs_Click(object sender, EventArgs e)
        {
            txtboxCurrentCalc.Text += ((System.Windows.Forms.Button)sender).Text;
        }

        private void btnPlus_Click(object sender, EventArgs e)
        {
            if (Decimal.TryParse(txtboxCurrentCalc.Text, out decimal displayValue))
            {
                calculator.Add(displayValue);

                textBoxHistoric.Text = calculator.Operand1.ToString() + " + ";
                txtboxCurrentCalc.Text = "";
            }
        }

        private void btnMinus_Click(object sender, EventArgs e)
        {
            if (Decimal.TryParse(txtboxCurrentCalc.Text, out decimal displayValue))
            {
                calculator.Subtract(displayValue);

                textBoxHistoric.Text = calculator.Operand1.ToString() + " - ";
                txtboxCurrentCalc.Text = "";
            }
        }

        private void btnMult_Click(object sender, EventArgs e)
        {
            if (Decimal.TryParse(txtboxCurrentCalc.Text, out decimal displayValue))
            {
                calculator.Multiply(displayValue);

                textBoxHistoric.Text = calculator.Operand1.ToString() + " * ";
                txtboxCurrentCalc.Text = "";
            }
        }

        private void btnDiv_Click(object sender, EventArgs e)
        {
            if (Decimal.TryParse(txtboxCurrentCalc.Text, out decimal displayValue))
            {
                calculator.Divide(displayValue);

                textBoxHistoric.Text = calculator.Operand1.ToString() + " / ";
                txtboxCurrentCalc.Text = "";
            }
        }

        private void btnEqual_Click(object sender, EventArgs e)
        {
            if (Decimal.TryParse(txtboxCurrentCalc.Text, out decimal displayValue))
            {
                calculator.EqualsOp(displayValue);
                textBoxHistoric.Text += (txtboxCurrentCalc.Text) + " = " +
calculator.CurrentValue.ToString();

                calculator.Clear();
                txtboxCurrentCalc.Text = "";


                // Write in Text File:
                FileStream fs = null;

                try
                {
                    fs = new FileStream(@".\Calculator.txt", FileMode.Append,
FileAccess.Write);

                    // Create the output stream for a text file that exists
                    StreamWriter textOut = new StreamWriter(fs);

                    // Write the following fields into text file:
                    textOut.WriteLine(textBoxHistoric.Text);
                    textOut.Close();
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                }
                finally
                {
                    if (fs != null) fs.Close();
                }

            }
        }

        private void btnClear_Click(object sender, EventArgs e)
        {
            calculator.Clear();
            txtboxCurrentCalc.Text = "";
            textBoxHistoric.Text = "";
        }

        private void btnExit_Click(object sender, EventArgs e)
        {
            if (MessageBox.Show("Do you want to quit this application? ", "Exit ?",
MessageBoxButtons.YesNo).ToString() == "Yes")
            {
                this.Close();
            }
        }
    }
```

IP VALIDATOR:

```
public partial class IP4validator : Form
    {
        public IP4validator()
        {
            InitializeComponent();
        }

        // Global variables
        string currentDateTime = DateTime.Now.ToLongDateString();
        DateTime openingTime;

        private void IP4validator_Load(object sender, EventArgs e)
        {
            // Display current date and time in the label
            label1.Text = currentDateTime;

            // Save the exactly hour when the form is loaded
            openingTime = DateTime.Now;
        }

        private void btnValidateIP_Click(object sender, EventArgs e)
        {
            string ipToValidate = textBox1.Text.Trim();

            Regex myRegex = new Regex(@"^(25[0-5]|2[0-4]\d|[0-1]?\d?\d)(\.(25[0-
5]|2[0-4]\d|[0-1]?\d?\d)){3}$");

            // Check if the input matches the regular expression
            bool isValid = myRegex.IsMatch(ipToValidate);

            if (isValid)
            {
                MessageBox.Show(ipToValidate + "\nThe IP is correct", "Valid IP");

                // Write the validated IP in a binary file:
                string bPath = @".\binaryValidIP.txt";

                FileStream fs = null;

                try
                {
                    fs = new FileStream(bPath, FileMode.Append, FileAccess.Write);

                    // create the output stream for a binary file that exists
                    BinaryWriter binaryOut = new BinaryWriter(fs);

                    // write the fields into text file
                    binaryOut.Write(textBox1.Text.Trim() + currentDateTime);

                    // close the output stream for the text file
                    binaryOut.Close();
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                }
                finally
                {
                    if (fs != null) fs.Close();
                }
            }

            else
            {
                MessageBox.Show(ipToValidate + "\nThe IP must have 4 bytes\ninteger
number between 0 to 255\nseparated by a dot (255.255.255.255)", "Error");
            }
        }

        private void btnResetIP_Click(object sender, EventArgs e)
        {
            textBox1.Text = "";

            textBox1.Focus();
        }

        private void btnExitIP_Click(object sender, EventArgs e)
        {
            if (MessageBox.Show("Do you want to quit this application? ", "Exit ?",
MessageBoxButtons.YesNo).ToString() == "Yes")
            {
                // Calculating the time spent between opening and closing the
application:
                TimeSpan usageTime = DateTime.Now - openingTime;

                // Show the time spent using the application:
                MessageBox.Show("The usage time of IP validator was:\n" +
usageTime.Minutes + " minutes and " + usageTime.Seconds + " seconds.", "Usage
Time");

                this.Close();
            }
        }
    }
```

D. Present the classes and/or methods that you create, or you did use in the project.

Class/Method Name	**Description**
`Currency`	This class was created to work on the MoneyEx Form, to reduce the number of event creation and to perform in a polished way the conversions between the presented currencies. So, it can always the Canadian dollar as a currency factor of conversion. It converts to CAD the value inserted and then convert it from CAD to the desired one.
`public double ConvertToCAD(double valueInserted) { return valueInserted * CanadianFactor; }`	This method is performing a conversion of the value inserted in the textbox to the Canadian currency factor.
`public double FromCADtoCurrency(double cadValue) { return cadValue/CanadianFactor; }`	This method is using the stored Canadian value of the return of ConvertToCAD and making the conversion to the desired currency.
Constructor()	`public Currency(double canadianFactor, string currencyName) { CanadianFactor = canadianFactor; CurrencyName = currencyName; }`

Class/Method Name	**Description**
`CalculatorClass`	This class is organizing the operands and operators so it can perform appropriately the calculation desired by the user. Below it's possible to see the methods that are inside:
`public void Add(decimal displayValue) { CurrentValue = displayValue; operand1 = CurrentValue; op = "+"; }`	Here this method stores the inserted value and the operation that is going to be realized, and here the addition (+). It also stores in another variable to show later.

`public void Subtract(decimal displayValue)`
`{`
`CurrentValue = displayValue;`
`operand1 = CurrentValue;`
`op = "-";`
`}`
	Here this method stores the inserted value and the operation that is going to be realized, and here the subtraction (-). It also stores in another variable to show later.

`public void Multiply(decimal displayValue)`
`{`
`CurrentValue = displayValue;`
`operand1 = CurrentValue;`
`op = "*";`
`}`
	Here this method stores the inserted value and the operation that is going to be realized, and here the multiplication (*). It also stores in another variable to show later.

`public void Divide(decimal displayValue)`
`{`
`CurrentValue = displayValue;`
`operand1 = CurrentValue;`
`op = "/";`
`}`
	Here this method stores the inserted value and the operation that is going to be realized, and here the division (/). It also stores in another variable to show later.

`public void EqualsOp(decimal displayValue)`
`{`
`Operand2 = displayValue;`
 	
`switch (op)`
`{`
`case "+":`
`CurrentValue = Operand1 +`
`Operand2;`
`break;`
 	
`case "-":`
`CurrentValue = Operand1 -`
`Operand2;`
`break;`
 	
`case "*":`
`CurrentValue = Operand1 *`
`Operand2;`
`break;`
 	
`case "/":`
`CurrentValue = Operand1 /`
`Operand2;`
`break;`
 	
`default:`
`break;`
`}`
`}`
	The calculation is realized here. After clicking in a number, store the number, click in an operation, store operation, select another number, and in the equal the calculation is concluded and displayed in the textbox “historic”.

`public void Clear() { Operand1 = 0; Operand2 = 0; CurrentValue = 0; op = null; }`	This method is just mean to clear the calculator as a whole.
Default Constructor()	`public CalculatorClass() { op = null; }`

E. Present the difficulties that you have, what was the hardest and the easiest part of your project.

I had difficulties understanding if I should create a separate class for the Calculator or it all code should have been inside the unique .cs file. I started doing without classes to check the logic behind this calculator. This happened in 10/04/2023.

On 14/04 I had issues while understanding what the correct value factor was to perform the conversions between the currencies. At first it looked right, then I realized it was not, and I was able to invert what was wrong, leading to the correct operations.

On 16/04 the calculator I was able to recreate the calculator with a class, as desired by the pdf instructions. This was, definitely, the most complex/hardest of the 6 applications. A lot of struggles to display properly the numbers and operators selected properly in the second textbox I called “historic”.

On 18/04, during the class, I have corrected my Regex for the IP application. Now it’s working properly fine. Regex myRegex = new Regex(@"^(25[0-5]|2[0-4]\d|[0-1]?\d?\d)(\.(25[0-5]|2[0-4]\d|[0-1]?\d?\d)){3}$");