

College LaSalle

# Project - Oriented Object Programming User and Technical Manual

Presented to: Mihai Maftai.

Your name: Bernadette Nicole Fernando  
4/4/2023  
Version: 2.0

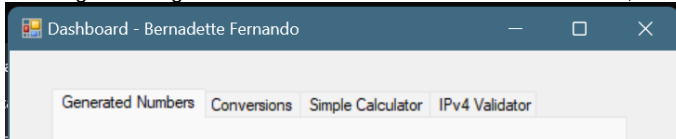
**A. Start by adding a short description of your project, and the languages (technologies) used:**

1. Language: C#
2. Tools (IDE):
  - o Microsoft Visual Studio Community 2022 Version 17.5.2
  - o Microsoft .NET Framework Version 4.8.09032
  - o Installed Version: Community

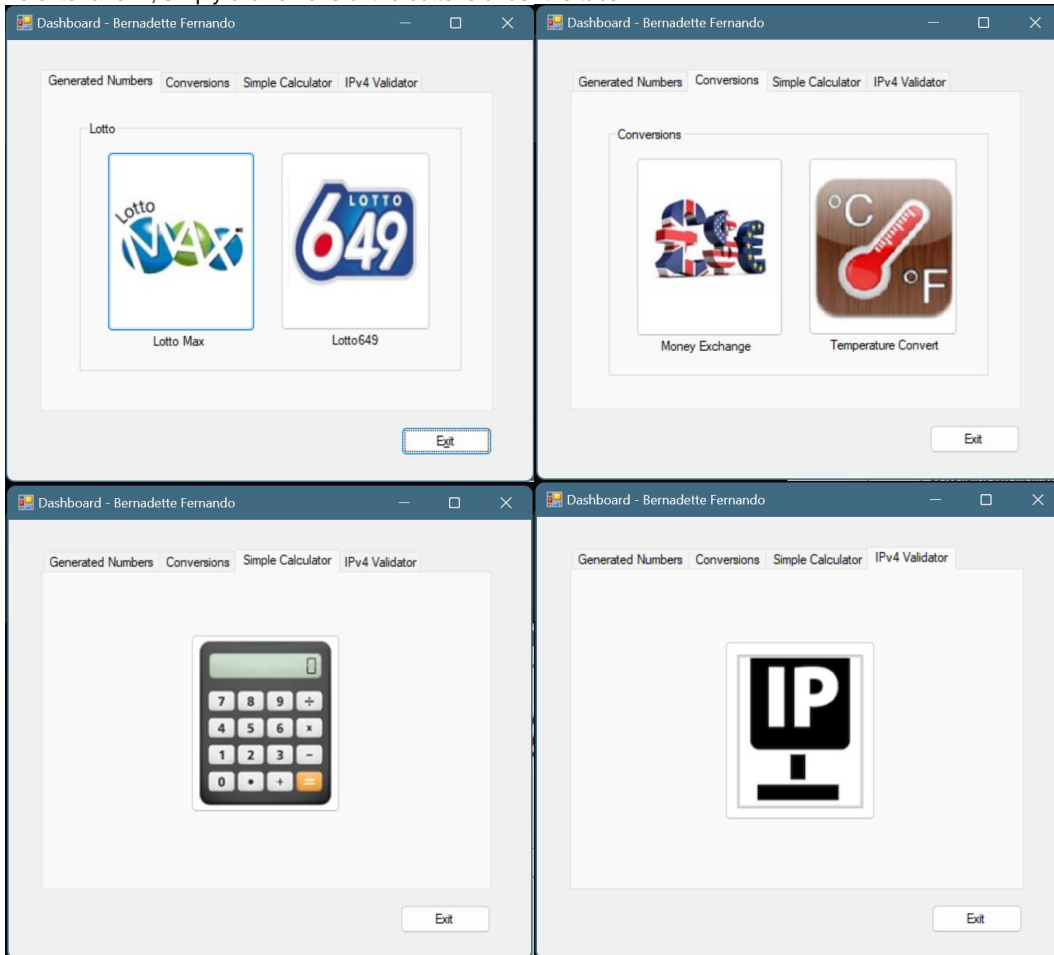
**B. Present the print screens of yours forms, and have a detailed description of the functionalities (step by step).**

**FORM:** DASHBOARD **NAME:** frmDashboard

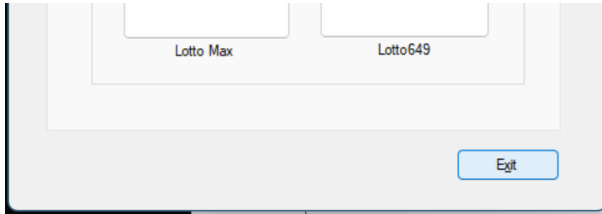
1. To navigate through the different forms available in the dashboard, click one of the tabs on top



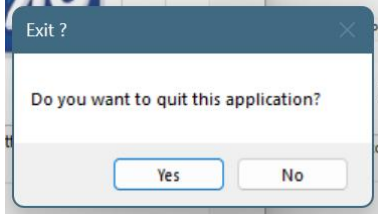
2. To enter a form, simply click on one of the buttons under the tabs



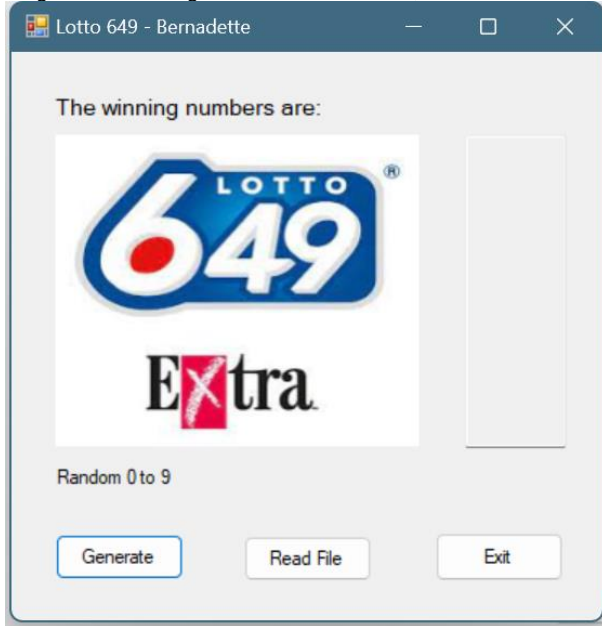
3. To exit the application, click the exit button at the bottom right of the dashboard



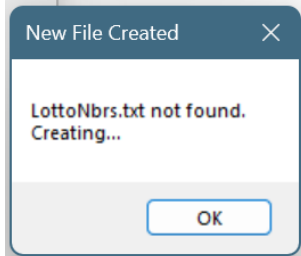
4. A prompt will appear and will require the user to confirm before closing the application



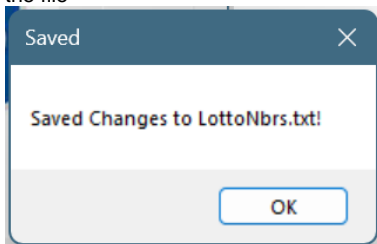
1. To generate winning numbers for Lotto 649, click the Generate button at the lower right part of the form



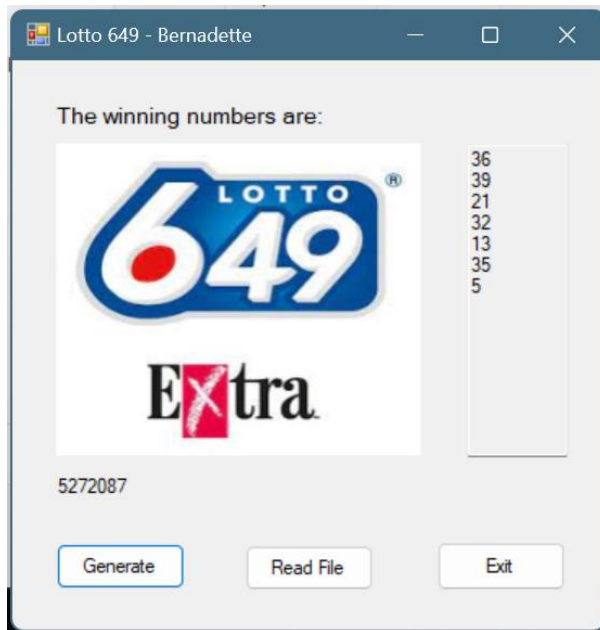
2. If the text file does not exist, a prompt will appear to tell the user that the text file is not found and will be created.



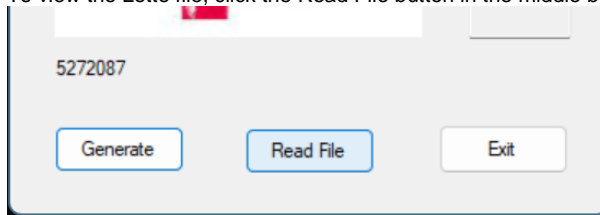
3. After the file is created, another prompt will appear to tell the user that the generated numbers are added successfully to the file



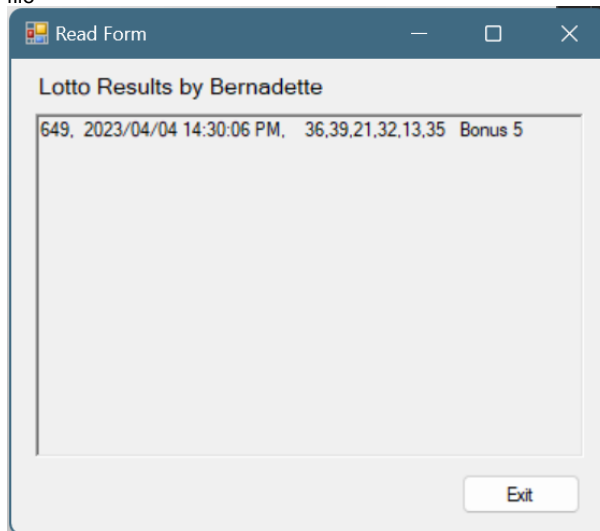
4. The results will be displayed on the form interface. The 6 winning numbers with 1 bonus are displayed on the read-only text box on the right and the random 7-digits are displayed on the bottom of the image



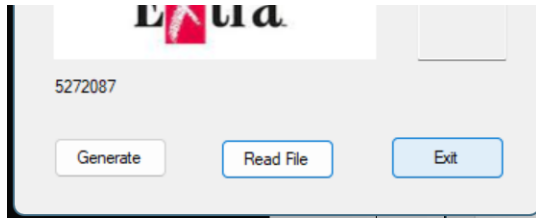
5. To view the Lotto file, click the Read File button in the middle below the form



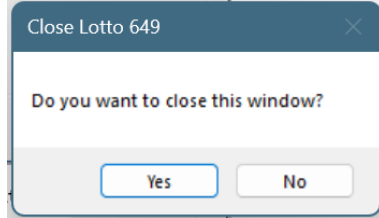
6. Clicking the Read File button will open up a form with a read-only multiline textbox that will display the content of the text file



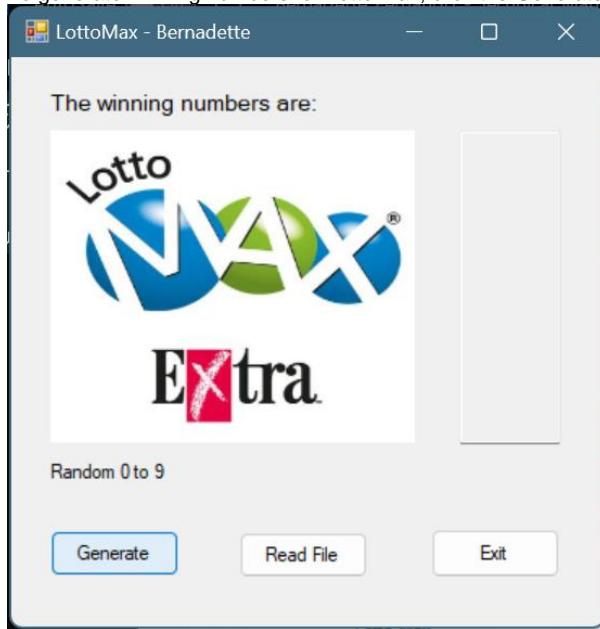
7. To exit form, click Exit button at the bottom right



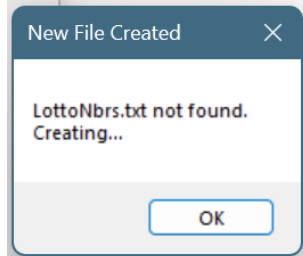
8. A prompt will appear asking for confirmation from user before the application closes



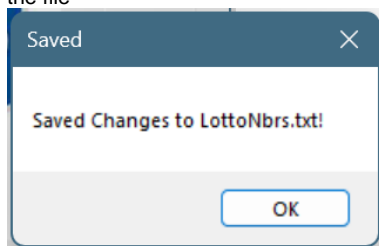
1. To generate winning numbers for Lotto Max, click the Generate button at the lower right part of the form



2. If the text file does not exist, a prompt will appear to tell the user that the text file is not found and will be created.



3. After the file is created, another prompt will appear to tell the user that the generated numbers are added successfully to the file



4. The results will be displayed on the form interface. The 7 winning numbers with 1 bonus are displayed on the read-only text box on the right and the random 7-digits are displayed on the bottom of the image

The winning numbers are:

**Lotto MAX Extra**

45  
3  
7  
33  
46  
25  
11  
47

5452000

Generate Read File Exit

- To view the Lotto file, click the Read File button in the middle below the form

5452000

Generate Read File Exit

- Clicking the Read File button will open up a form with a read-only multiline textbox that will display the content of the text file

Lotto Results by Bernadette

649, 2023/04/04 14:30:06 PM, 36,39,21,32,13,35 Bonus 5  
Max, 2023/04/04 14:31:51 PM, 45,3,7,33,46,25,11 Bonus 47

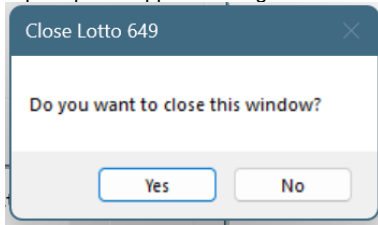
Exit

- To exit form, click Exit button at the bottom right

Generate Read File Exit



8. A prompt will appear asking for confirmation from user before the application closes



**FORM: MONEY EXCHANGE NAME: frmMoneyEx**

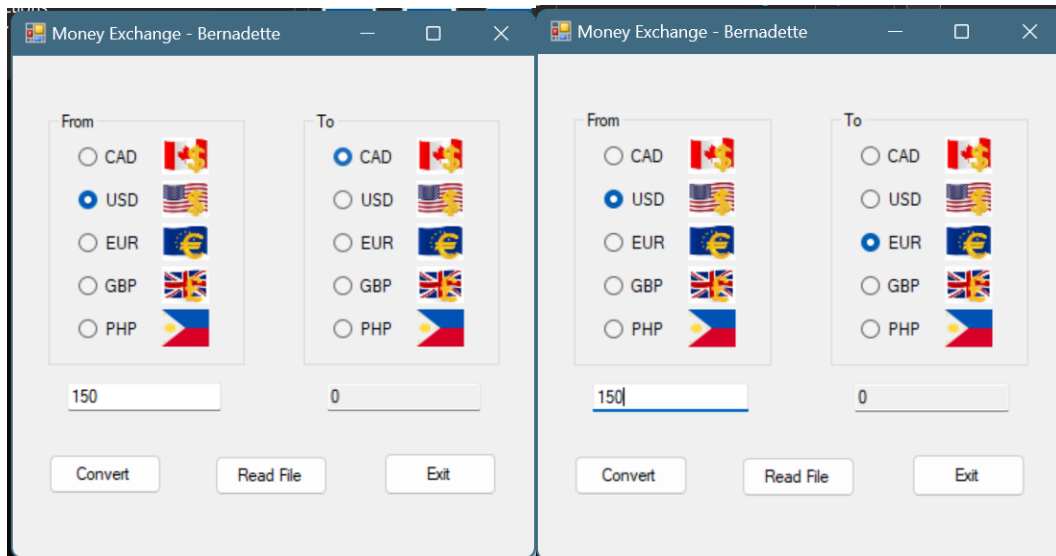
1. To convert to and from any currency, the user should select the radio button beside the currency to convert FROM (by default, it is CAD)

The image shows two side-by-side screenshots of the 'Money Exchange - Bernadette' application window. In both screenshots, the 'From' group box has 'CAD' selected with a radio button. The 'To' group box also has 'CAD' selected. The 'From' and 'To' textboxes both contain the number '0'. The 'Convert', 'Read File', and 'Exit' buttons are visible at the bottom.

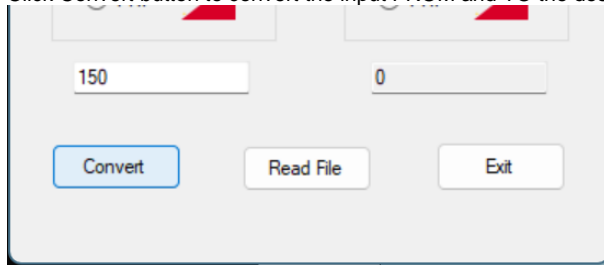
2. Input the amount to convert in the textbox under the FROM group box

The image shows a screenshot of the 'Money Exchange - Bernadette' application window. The 'From' group box has 'USD' selected with a radio button. The 'To' group box has 'CAD' selected with a radio button. The 'From' textbox contains the number '150' and the 'To' textbox contains the number '0'. The 'Convert', 'Read File', and 'Exit' buttons are visible at the bottom.

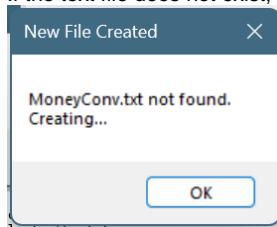
3. Next, the user should click the radio button beside the currency to convert TO (by default, it is CAD)



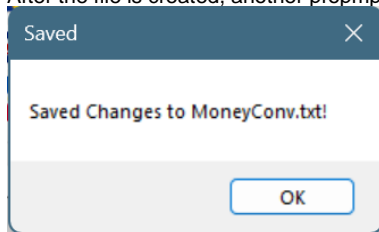
4. Click Convert button to convert the input FROM and TO the desired currencies



5. If the text file does not exist, a prompt will appear to tell the user that the text file is not found and will be created.



6. After the file is created, another prompt will appear to tell the user that the output is added successfully to the file

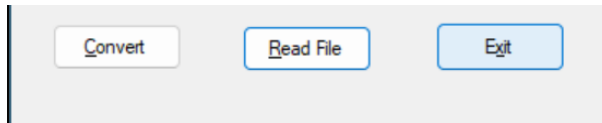


7. The results will be displayed on the form interface. The converted currency will be displayed on the read-only textbox under the TO group box

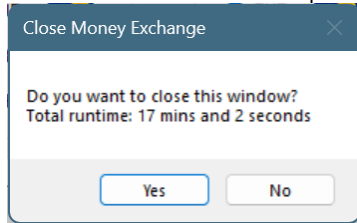
8. To view the content of the text file, click the Read File button in the middle below the form

9. Clicking the Read File button will open up a form with a read-only multiline textbox that will display the content of the text file

10. To exit form, click Exit button at the bottom right

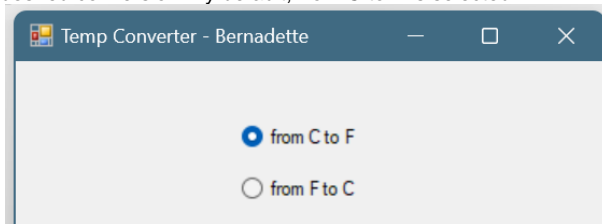


11. A prompt will appear asking for confirmation from user before the application closes. It will also display the total runtime of the form between the time it is opened and Exit button is clicked.



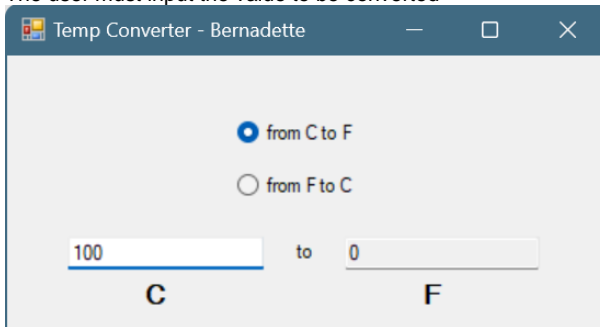
**FORM: TEMPERATURE CONVERT** **NAME:** frmTempConv

1. To convert temperature, the user must select between two conversion options: from C to F (converting from Celsius to Fahrenheit) or from F to C (converting from Fahrenheit to Celsius). The user should click the radio button beside the desired conversion. By default, from C to F is selected.



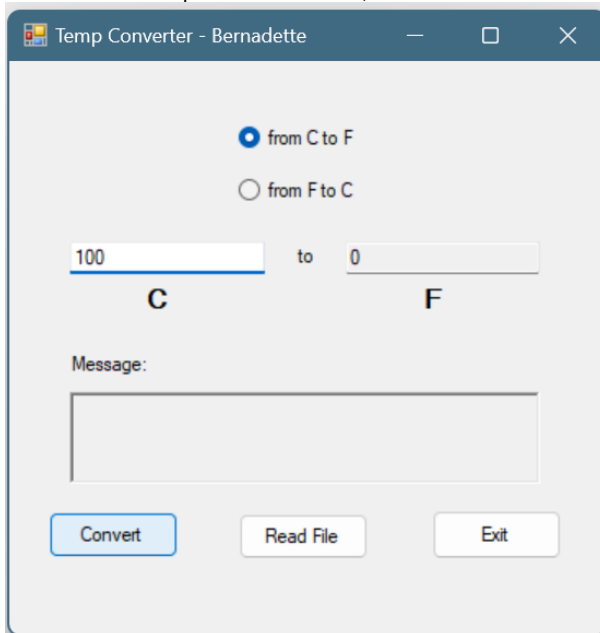
The screenshot shows a window titled "Temp Converter - Bernadette". Inside, there are two radio buttons. The first radio button is selected and is labeled "from C to F". The second radio button is unselected and is labeled "from F to C".

2. The user must input the value to be converted



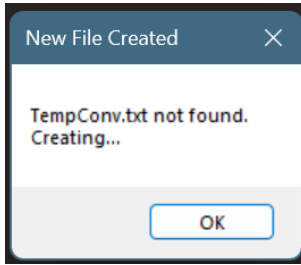
The screenshot shows the same window as before, but now with two input fields. The first input field contains the number "100" and is labeled "C" below it. The second input field contains the number "0" and is labeled "F" below it. The text "to" is placed between the two input fields. The radio buttons remain the same.

3. To convert the temperature on the left, click the Convert button on the lower left bottom of the form

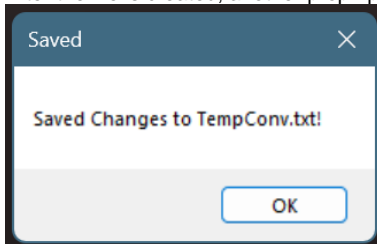


The screenshot shows the same window as before, but now with a "Convert" button at the bottom left. Below the input fields, there is a label "Message:" followed by a text area. At the bottom of the window, there are three buttons: "Convert", "Read File", and "Exit".

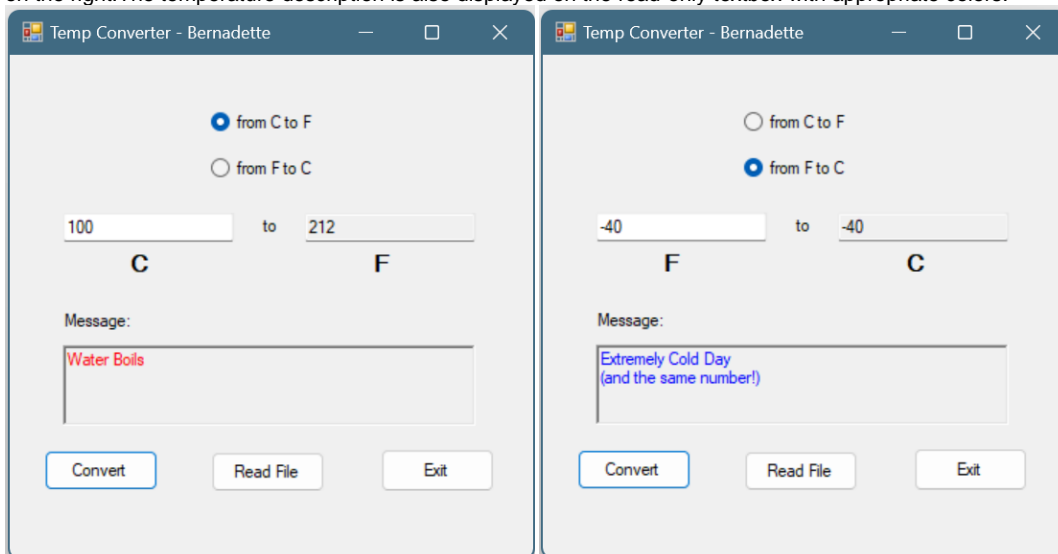
4. If the text file does not exist, a prompt will appear to tell the user that the text file is not found and will be created.



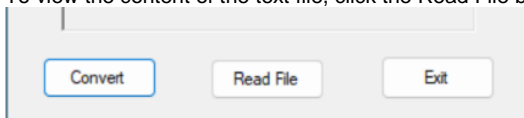
5. After the file is created, another prompt will appear to tell the user that the output is added successfully to the file



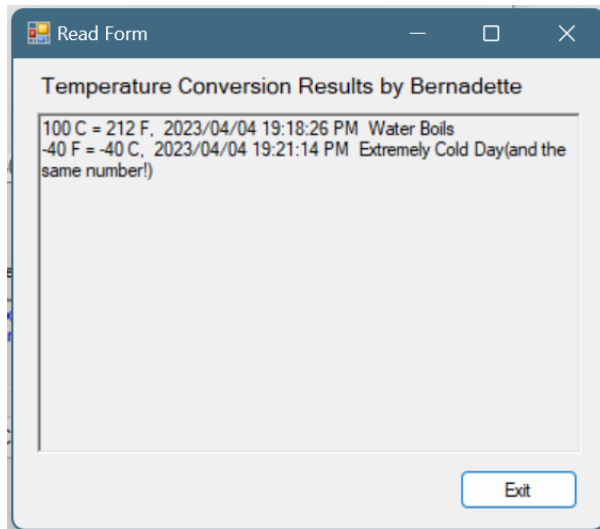
6. The results will be displayed on the form interface. The converted temperature will be displayed on the read-only textbox on the right. The temperature description is also displayed on the read-only textbox with appropriate colors.



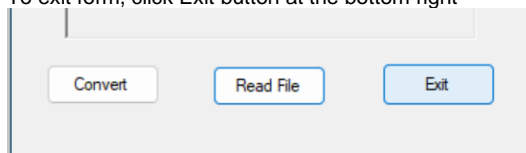
7. To view the content of the text file, click the Read File button in the middle below the form



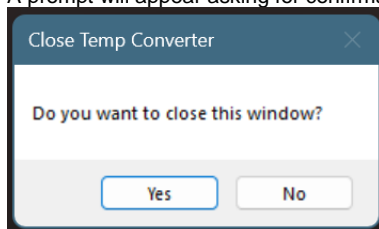
8. Clicking the Read File button will open up a form with a read-only multiline textbox that will display the content of the text file



9. To exit form, click Exit button at the bottom right

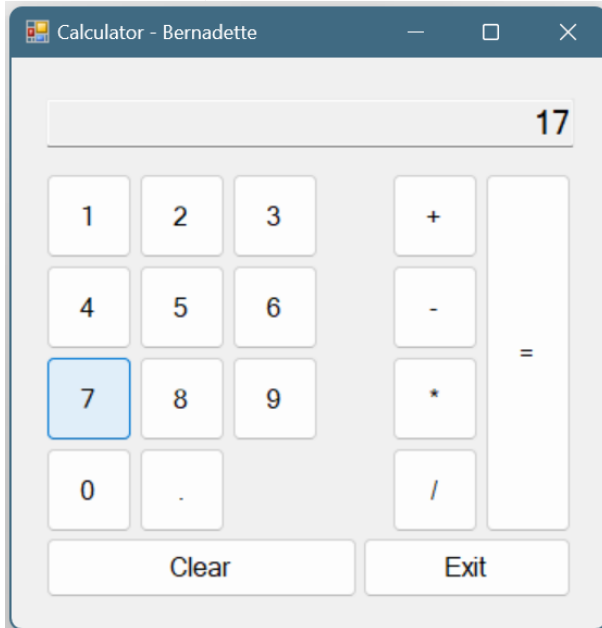


10. A prompt will appear asking for confirmation from user before the application closes.

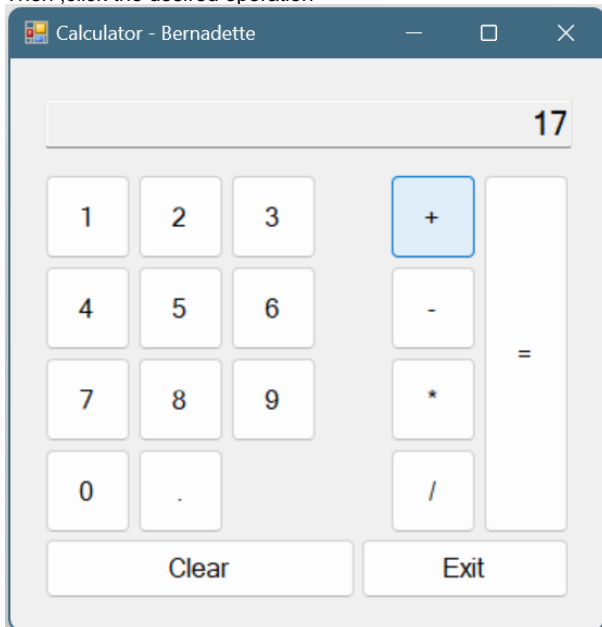




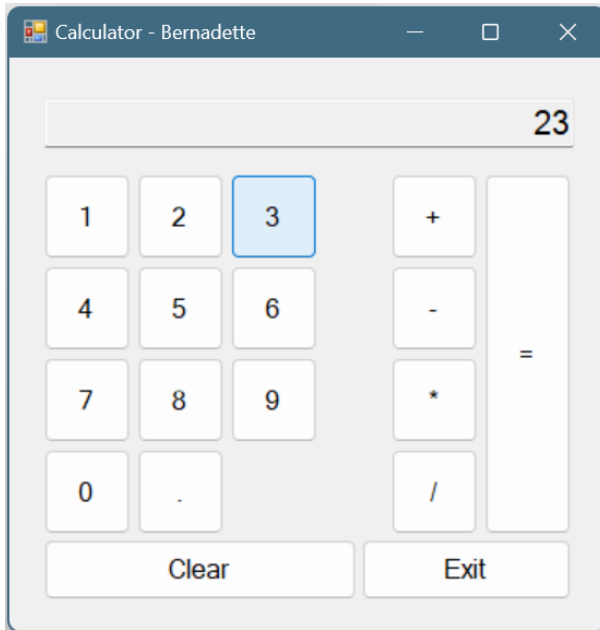
1. To use the calculator, click the number buttons to build the first number



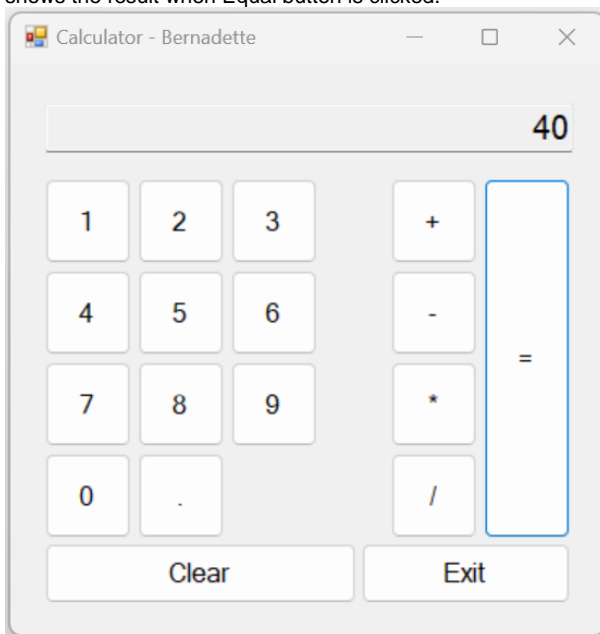
2. Then ,click the desired operation



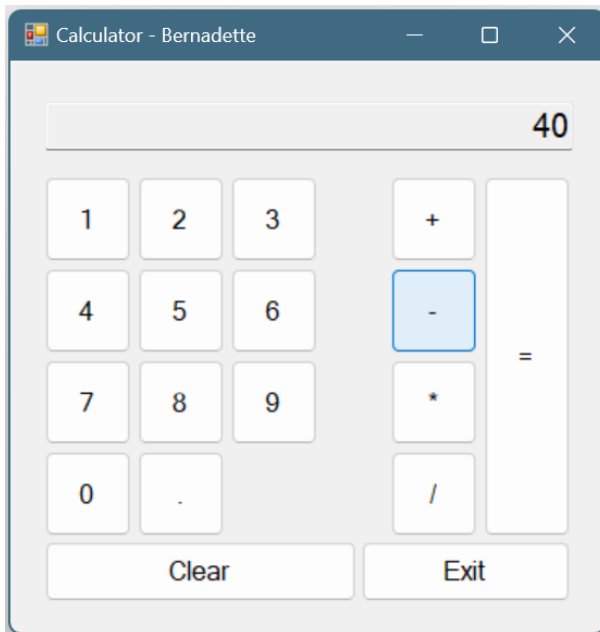
3. Then, click the number buttons to build the second number



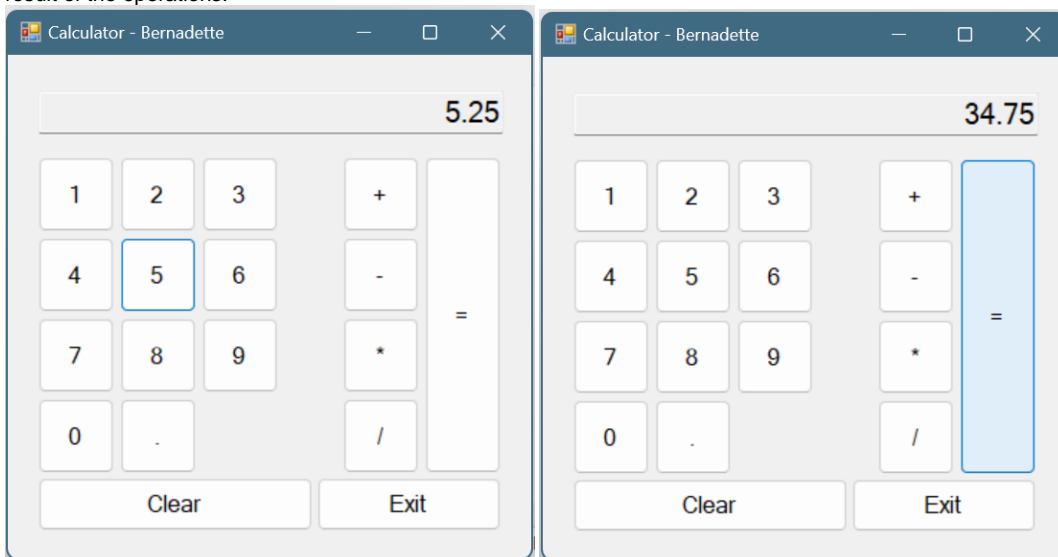
4. To view the result, there can be two ways: Clicking the equal button or clicking the next operation. Screenshot below shows the result when Equal button is clicked.



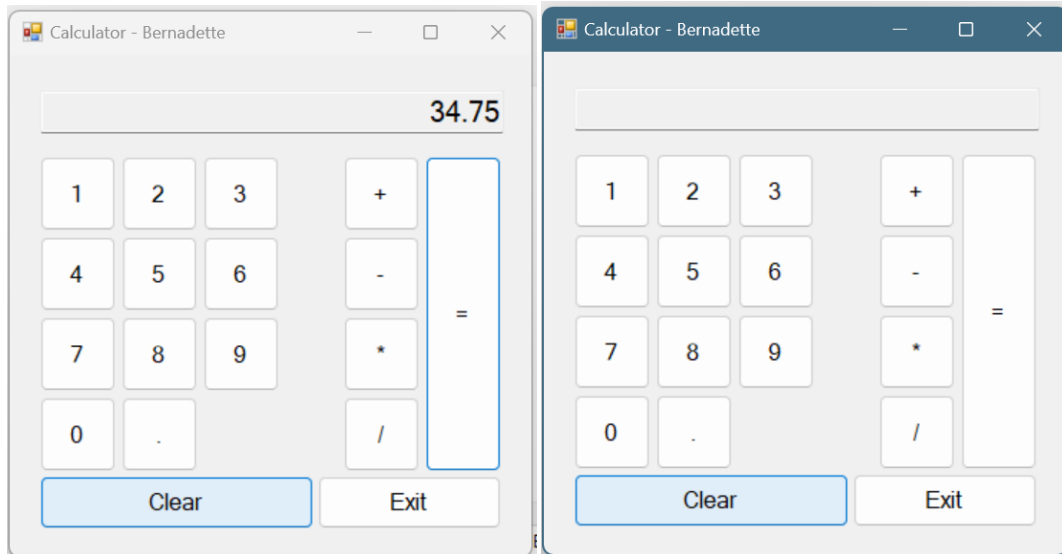
5. When the user clicks the next operation, the result of the previous two numbers will be the first number and the user can once again click the number buttons for the second number. Screenshot below shows the result when Operator button is clicked.



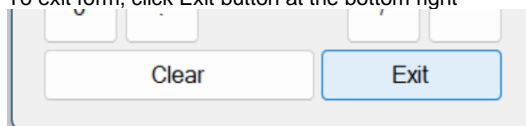
6. After the second number is entered, clicking any of the buttons (+,-,\*,/,=) on the right of the number buttons will display the result of the operations.



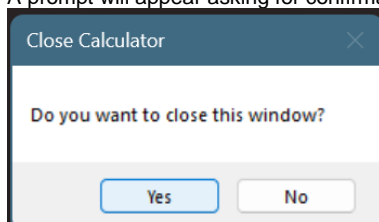
7. To clear or reset the calculator, click the Clear button on the lower left of the form



8. To exit form, click Exit button at the bottom right

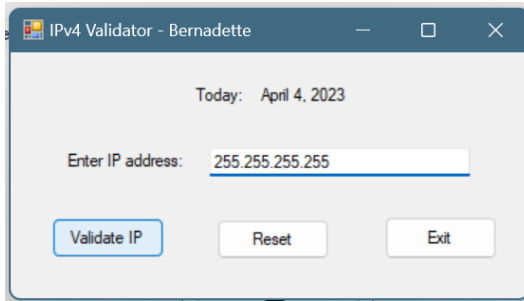


9. A prompt will appear asking for confirmation from user before the application closes.



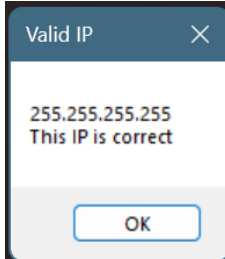
**FORM: IPV4 VALIDATOR NAME: frmIPv4Validator**

1. To validate an IP address, type a valid IPv4 address on the textbox then click the Validate button on the lower left side of the form



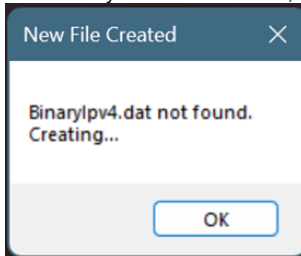
The screenshot shows a window titled "IPv4 Validator - Bernadette". At the top, it says "Today: April 4, 2023". Below that is a label "Enter IP address:" followed by a text box containing "255.255.255.255". At the bottom, there are three buttons: "Validate IP", "Reset", and "Exit".

2. If it's a valid IPv4 address, a prompt confirming the entry will appear.



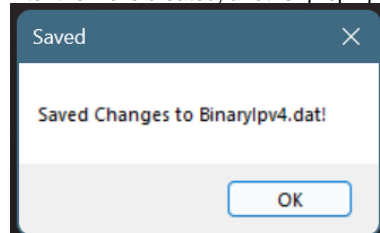
The screenshot shows a small dialog box titled "Valid IP" with a close button (X). The text inside says "255.255.255.255" and "This IP is correct". At the bottom is an "OK" button.

3. If the binary file does not exist, a prompt will appear to tell the user that the binary file is not found and will be created.



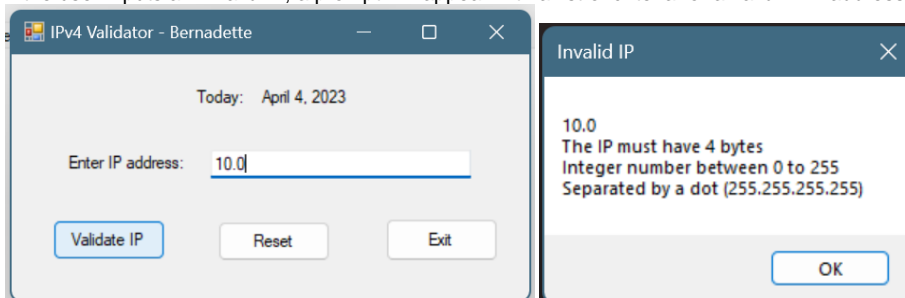
The screenshot shows a small dialog box titled "New File Created" with a close button (X). The text inside says "Binary\ipv4.dat not found." and "Creating...". At the bottom is an "OK" button.

4. After the file is created, another prompt will appear to tell the user that the output is added successfully to the file



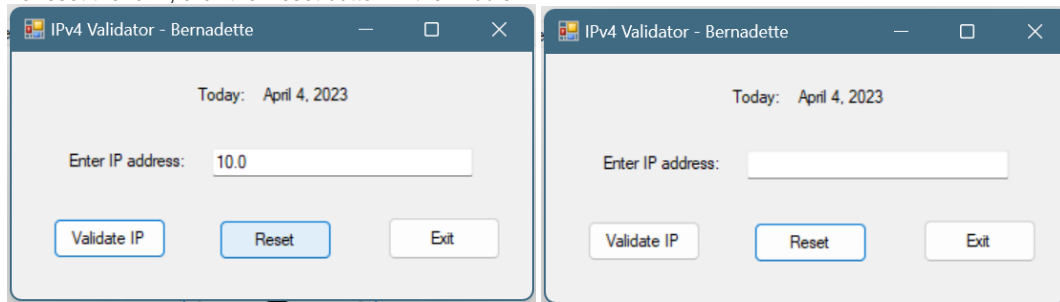
The screenshot shows a small dialog box titled "Saved" with a close button (X). The text inside says "Saved Changes to Binary\ipv4.dat!". At the bottom is an "OK" button.

5. If the user inputs an invalid IP, a prompt will appear with a list of criteria for a valid IPv4 address

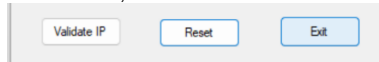


The screenshot shows two windows. On the left is the "IPv4 Validator - Bernadette" window with "10.0" entered in the text box. On the right is a dialog box titled "Invalid IP" with a close button (X). The text inside lists criteria: "10.0", "The IP must have 4 bytes", "Integer number between 0 to 255", and "Separated by a dot (255.255.255.255)". At the bottom is an "OK" button.

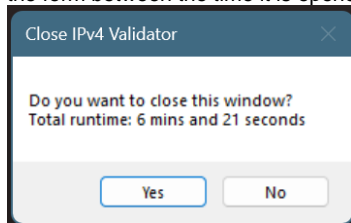
6. To reset the form, click the Reset button in the middle



7. To exit form, click Exit button at the bottom right

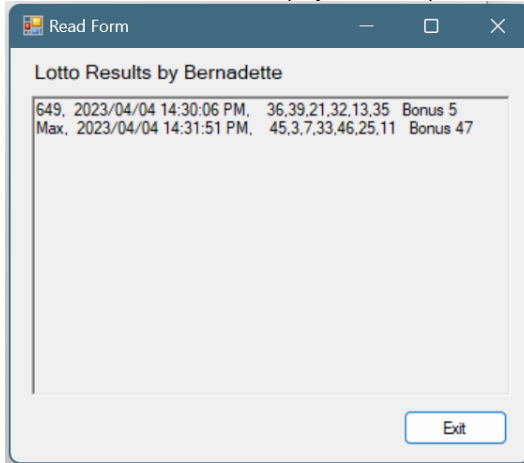


8. A prompt will appear asking for confirmation from user before the application closes. It will also display the total runtime of the form between the time it is opened and Exit button is clicked.



**FORM: DISPLAY READ FILE NAME: frmReadFile**

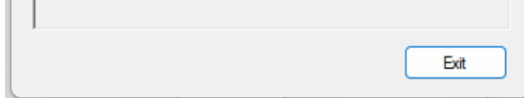
1. The read form serves as a display for the outputs of the other forms. It displays the output on a read-only textbox



The screenshot shows a Windows-style window titled "Read Form". Inside the window, there is a label "Lotto Results by Bernadette" above a read-only text area. The text area contains two lines of data: "649, 2023/04/04 14:30:06 PM, 36,39,21,32,13,35 Bonus 5" and "Max, 2023/04/04 14:31:51 PM, 45,3,7,33,46,25,11 Bonus 47". At the bottom right of the window is an "Exit" button.

Lotto Results by Bernadette			
649,	2023/04/04 14:30:06 PM,	36,39,21,32,13,35	Bonus 5
Max,	2023/04/04 14:31:51 PM,	45,3,7,33,46,25,11	Bonus 47

2. To close the form, simply click the Exit button on the lower right



This is a close-up view of the bottom right corner of the "Read Form" window, showing the "Exit" button.

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

FORM: DASHBOARD NAME: frmDashboard

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

// Bernadette Fernando
// April 04 2023
// This windows form will provide access to different forms that are able to do the following:
// Generate Numbers, Convert Currency, Calculator, Validate IPv4

namespace WinFormProject
{
    public partial class frmDashboard : Form
    {
        public frmDashboard()
        {
            InitializeComponent();
        }

        private void btnLottoMax_Click(object sender, EventArgs e)
        {
            frmMax app = new frmMax();
            app.Show();
        }

        private void btnLotto649_Click(object sender, EventArgs e)
        {
            frm649 app = new frm649();
            app.Show();
        }

        private void btnMoneyEx_Click(object sender, EventArgs e)
        {
            frmMoneyEx app = new frmMoneyEx();
            app.Show();
        }

        private void btnTempConv_Click(object sender, EventArgs e)
        {
            frmTempConv app = new frmTempConv();
            app.Show();
        }

        private void btnCalc_Click(object sender, EventArgs e)
        {
            frmCalculator app = new frmCalculator();
            app.Show();
        }

        private void btnIPVal_Click(object sender, EventArgs e)
        {
            frmIPv4Validator app = new frmIPv4Validator();
            app.Show();
        }

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



```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WinFormProject
{
    public partial class frm649 : Form
    {
        public frm649()
        {
            InitializeComponent();
        }

        private void btnL649Gen_Click(object sender, EventArgs e)
        {
            // BASIC FUNCTION
            // clear the textbox if button is clicked
            txtL649.Clear();

            var generatedNum = new List<int>();
            Random random = new Random();

            // generate random numbers from 0 to 9
            string randomNumber = random.Next(0, 9).ToString();
            for (int i = 0; i < 6; i++)
            {
                randomNumber += random.Next(0, 9).ToString();
            }

            lbl64909.Text = randomNumber;

            // generate first number and add to list
            int firstRNum = random.Next(1, 49);
            generatedNum.Add(firstRNum);

            // generate a new number and add to list
            int newRNum;
            for (int i = 0; i < 6; i++)
            {
                // generate a new number while it exists in the list
                // validation to ensure that every number is unique
                do
                {
                    newRNum = random.Next(1, 49);
                } while (generatedNum.Contains(newRNum));

                generatedNum.Add(newRNum);
            }

            // store the last generated number as bonus
            var lastItem = generatedNum.Last();

            // display list
            foreach (int num in generatedNum)
            {
                txtL649.Text += num.ToString() + "\r\n";
            }

            // WRITE TO TEXT FILE
            string outputDesc = $"{",",-5}";
            for (int i = 0; i < 6; i++)
            {
                // position the commas
                outputDesc += generatedNum[i];
                if (i < 5)
                {
                    outputDesc += ",";
                }
            }
        }
    }
}

```

```

    }
}

outputDesc += $"{"Bonus",8} {lastItem}";

DataStream toWrite = new DataStream();

toWrite.FileName = "LottoNbrs";
toWrite.MsgBoxTitle = "Lotto";
toWrite.Output = "649";
toWrite.Description = outputDesc;

toWrite.WriteFile();
}

private void btn649Exit_Click(object sender, EventArgs e)
{
    if (MessageBox.Show("Do you want to close this window? ", "Close Lotto 649", MessageBoxButtons.YesNo)
== DialogResult.Yes)
    {
        this.Close();
    }
}

private void btnL649Read_Click(object sender, EventArgs e)
{
    // READ TEXT FILE
    DataStream toRead = new DataStream();
    toRead.FileName = "LottoNbrs";
    toRead.MsgBoxTitle = "Lotto649";

    frmReadFile readDisplay = new frmReadFile();
    readDisplay.fileOutput = toRead.ReadFile();
    readDisplay.frmTitle = toRead.MsgBoxTitle;
    readDisplay.Show();
}

private void frm649_Load(object sender, EventArgs e)
{
}
}
}

```

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Reflection;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

// Bernadette Fernando
// Windows Form Project
// April 18, 2023

namespace WinFormProject
{
    public partial class frmMax : Form
    {
        public frmMax()
        {
            InitializeComponent();

            private void btnLMaxGen_Click(object sender, EventArgs e)
            {
                // BASIC FUNCTION
                // clear the textbox if button is clicked
                txtLMax.Clear();

                var generatedNum = new List<int>();
                Random random = new Random();

                // generate random numbers from 0 to 9
                string randomNumber = random.Next(0, 9).ToString();
                for (int i = 0; i < 6; i++)
                {
                    randomNumber += random.Next(0, 9).ToString();
                }

                lblLMax09.Text = randomNumber;

                // generate first number and add to list
                int firstRNum = random.Next(1, 50);
                generatedNum.Add(firstRNum);

                // generate a new number and add to list
                int newRNum;
                for (int i = 0; i < 7; i++)
                {
                    // generate a new number while it exists in the list
                    // validation to ensure that every number is unique
                    do
                    {
                        newRNum = random.Next(1, 50);
                    } while (generatedNum.Contains(newRNum));

                    generatedNum.Add(newRNum);
                }

                // store the last generated number as bonus
                var lastItem = generatedNum.Last();

                foreach (int num in generatedNum)
                {
                    txtLMax.Text += num.ToString() + "\r\n";
                }

                // WRITE TO TEXT FILE
                string outputDesc = $"{"",-5}";
                for (int i = 0; i < 7; i++)
                {

```

```

        // position the commas
        outputDesc += generatedNum[i];
        if (i < 6)
        {
            outputDesc += ",";
        }
    }

    outputDesc += $"{"Bonus",8} {lastItem}";

    DataStream toWrite = new DataStream();

    toWrite.FileName = "LottoNbrs";
    toWrite.MsgBoxTitle = "Lotto";
    toWrite.Output = "Max";
    toWrite.Description = outputDesc;

    toWrite.WriteFile();
}

private void btnLMaxExit_Click(object sender, EventArgs e)
{
    if (MessageBox.Show("Do you want to close this window? ", "Close LottoMax", MessageBoxButtons.YesNo)
== DialogResult.Yes)
    {
        this.Close();
    }
}

private void btnLMaxRead_Click(object sender, EventArgs e)
{
    // READ TEXT FILE
    DataStream toRead = new DataStream();
    toRead.FileName = "LottoNbrs";
    toRead.MsgBoxTitle = "LottoMax";

    frmReadFile readDisplay = new frmReadFile();
    readDisplay.fileOutput = toRead.ReadFile();
    readDisplay.frmTitle = toRead.MsgBoxTitle;
    readDisplay.Show();
}
}
}

```

```

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

// March 16, 2023
// Money Conversion Factors (based on 1CAD):
// Source: Royal Bank of Canada
// 0.71230 USD
// 0.66609 EUR
// 0.58333 GBP
// 38.22630 PHP

namespace WinFormProject
{
    public partial class frmMoneyEx : Form
    {
        public frmMoneyEx()
        {
            InitializeComponent();
        }

        DateTime openFormTime, closeFormTime;
        string currencyFrom, currencyTo;

        private void btnMEX_Click(object sender, EventArgs e)
        {
            // BASIC FUNCTION
            CurrencyEx moneyConvert = new CurrencyEx();

            // FROM Group
            if (optUSD.Checked)
            {
                currencyFrom = "USD";
            }
            else if (optEUR.Checked)
            {
                currencyFrom = "EUR";
            }
            else if (optGBP.Checked)
            {
                currencyFrom = "GBP";
            }
            else if (optPHP.Checked)
            {
                currencyFrom = "PHP";
            }
            else if (optCAD.Checked)
            {
                currencyFrom = "CAD";
            }

            // TO Group
            if (optToUSD.Checked)
            {
                currencyTo = "USD";
            }
            else if (optToEUR.Checked)
            {
                currencyTo = "EUR";
            }
            else if (optToGBP.Checked)
            {
                currencyTo = "GBP";
            }
            else if (optToPHP.Checked)
            {
                currencyTo = "PHP";
            }
        }
    }
}

```

```

else if (optToCAD.Checked)
{
    currencyTo = "CAD";
}

try
{
    moneyConvert.Amt = Convert.ToDecimal(txtConvFrom.Text.Trim());

    // convert amount to CAD
    decimal convCAD = moneyConvert.ConvertToCAD(currencyFrom);

    // convert CAD to any amount
    decimal convCurrency = moneyConvert.ConvertToCurrency(convCAD, currencyTo);
    txtConvTo.Text = convCurrency.ToString();

    // WRITE TO TEXT FILE
    StreamWriter toWrite = new StreamWriter();

    toWrite.FileName = "MoneyConv";
    toWrite.MsgBoxTitle = "Money Conversion";
    toWrite.Output = $"{moneyConvert.Amt} {currencyFrom} = {convCurrency} {currencyTo}";
    toWrite.Description = "";

    toWrite.WriteLine();

}
catch (Exception err)
{
    MessageBox.Show($"Error! {err.Message}", "Exception Error");
    txtConvFrom.Text = "0";
    txtConvFrom.Focus();
}
}

private void btnMExRead_Click(object sender, EventArgs e)
{
    // READ TEXT FILE
    StreamReader toRead = new StreamReader();
    toRead.FileName = "MoneyConv";
    toRead.MsgBoxTitle = "Money Conversion";

    frmReadFile readDisplay = new frmReadFile();
    readDisplay.fileOutput = toRead.ReadFile();
    readDisplay.frmTitle = toRead.MsgBoxTitle;
    readDisplay.Show();
}

private void frmMoneyEx_Load(object sender, EventArgs e)
{
    optCAD.Checked = true;
    optToCAD.Checked = true;
    txtConvFrom.Text = "0";
    txtConvTo.Text = "0";
    openFormTime = DateTime.Now;
}

private void btnMExExit_Click(object sender, EventArgs e)
{
    closeFormTime = DateTime.Now;
    TimeSpan totRunTime = closeFormTime.Subtract(openFormTime);

    if (MessageBox.Show("Do you want to close this window?\n" +
        $"Total runtime: {totRunTime.Minutes} mins and {totRunTime.Seconds} seconds", "Close Money
Exchange", MessageBoxButtons.YesNo) == DialogResult.Yes)
    {
        this.Close();
    }
}
}
}

```

```

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

namespace WinFormProject
{
    public partial class frmTempConv : Form
    {
        public frmTempConv()
        {
            InitializeComponent();
        }

        private void btnTConv_Click(object sender, EventArgs e)
        {
            ConvertTemp tempConvert = new ConvertTemp();
            decimal convTemp;

            try
            {
                // BASIC FUNCTION
                string inputTemp = txtTempFrom.Text.Trim();
                tempConvert.Temp = Convert.ToDecimal(inputTemp);

                char unitFrom, unitTo;
                if (optCtoF.Checked)
                {
                    // convert temp from C to F
                    convTemp = tempConvert.ConvertToFahrenheit();
                    txtTempTo.Text = convTemp.ToString();
                    unitFrom = 'C';
                    unitTo = 'F';

                    // change text color
                    if (tempConvert.Temp >= 40)
                    {
                        rtfTempDesc.ForeColor = Color.Red;
                    }
                    else if (tempConvert.Temp >= 21 && tempConvert.Temp < 40)
                    {
                        rtfTempDesc.ForeColor = Color.Green;
                    }
                    else if (tempConvert.Temp < 21)
                    {
                        rtfTempDesc.ForeColor = Color.Blue;
                    }
                }
            }
            else
            {
                // convert temp from F to C
                convTemp = tempConvert.ConvertToCelsius();
                txtTempTo.Text = convTemp.ToString();
                unitFrom = 'F';
                unitTo = 'C';

                // change text color
                if (tempConvert.Temp >= 104)
                {
                    rtfTempDesc.ForeColor = Color.Red;
                }
                else if (tempConvert.Temp >= 70 && tempConvert.Temp < 104)
                {
                    rtfTempDesc.ForeColor = Color.Green;
                }
                else if (tempConvert.Temp < 70)
                {
                    rtfTempDesc.ForeColor = Color.Blue;
                }
            }
        }
    }
}

```

```

        // display temp description
        string tempDescription = tempConvert.DescribeTemperature(unitFrom);
        rtfTempDesc.Text = tempDescription;

        // WRITE TO TEXT FILE
        DataStream toWrite = new DataStream();

        toWrite.FileName = "TempConv";
        toWrite.MsgBoxTitle = "Temperature Conversion";
        toWrite.Output = $"{inputTemp} {unitFrom} = {convTemp} {unitTo}";

        // remove line break before saving to text file
        tempDescription = tempDescription.Replace("\n", "").Replace("\r", " ");
        toWrite.Description = " " + tempDescription;

        toWrite.WriteFile();
    }
    catch (Exception err)
    {
        MessageBox.Show($"Error! {err.Message}", "Exception Error");
        txtTempFrom.Text = "0";
        rtfTempDesc.Text = "";
        txtTempFrom.Focus();
    }
}

private void optFtoC_CheckedChanged(object sender, EventArgs e)
{
    lblFrom.Text = "F";
    lblTo.Text = "C";
}

private void optCtoF_CheckedChanged(object sender, EventArgs e)
{
    lblFrom.Text = "C";
    lblTo.Text = "F";
}

private void frmTempConv_Load(object sender, EventArgs e)
{
    optCtoF.Checked = true;
    optFtoC.Checked = false;
    txtTempFrom.Text = "0";
    txtTempTo.Text = "0";
    rtfTempDesc.Text = "";
}

private void btnTConvRead_Click(object sender, EventArgs e)
{
    // READ TEXT FILE
    DataStream toRead = new DataStream();
    toRead.FileName = "TempConv";
    toRead.MsgBoxTitle = "Temperature Conversion";

    frmReadFile readDisplay = new frmReadFile();
    readDisplay.fileOutput = toRead.ReadFile();
    readDisplay.frmTitle = toRead.MsgBoxTitle;
    readDisplay.Show();
}

private void btnTConvExit_Click(object sender, EventArgs e)
{
    if (MessageBox.Show("Do you want to close this window? ", "Close Temp Converter",
        MessageBoxButtons.YesNo) == DialogResult.Yes)
    {
        this.Close();
    }
}
}

```



```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static System.Windows.Forms.VisualStyles.VisualStyleElement.ProgressBar;

namespace WinFormProject
{
    public partial class frmCalculator : Form
    {
        public frmCalculator()
        {
            InitializeComponent();

        }

        private void frmCalculator_Load(object sender, EventArgs e)
        {
            toWrite.MsgBoxTitle = "Calculator";
        }

        // initiate constructors
        // Calculator - handle calculations
        Calculator calculate = new Calculator();
        // Datastream - handle read and write to text file
        DataStream toWrite = new DataStream();

        // variable to build the number
        private string numBuilder = "";

        // list that will hold the number inputs and results
        public List<double> listNumbers = new List<double>();

        // set the state of the buttons
        private bool btnAddClicked = false;
        private bool btnSubClicked = false;
        private bool btnMulClicked = false;
        private bool btnDivClicked = false;
        private bool btnEqualClicked = false;
        private bool btnDecClicked = false;
        double res;

        // number buttons
        private void btn1_Click(object sender, EventArgs e)
        {
            // if equal btn is already clicked, reset the string before input of new number
            if (btnEqualClicked)
            {
                numBuilder = "";
                btnEqualClicked = false;
            }

            numBuilder += "1";
            txtCalc.Text = numBuilder;
        }

        private void btn2_Click(object sender, EventArgs e)
        {
            if (btnEqualClicked)
            {
                numBuilder = "";
                btnEqualClicked = false;
            }
            numBuilder += "2";
            txtCalc.Text = numBuilder;
        }

        private void btn3_Click(object sender, EventArgs e)
        {

```

```

        if (btnEqualClicked)
        {
            numBuilder = "";
            btnEqualClicked = false;
        }
        numBuilder += "3";
        txtCalc.Text = numBuilder;
    }

    private void btn4_Click(object sender, EventArgs e)
    {
        if (btnEqualClicked)
        {
            numBuilder = "";
            btnEqualClicked = false;
        }
        numBuilder += "4";
        txtCalc.Text = numBuilder;
    }

    private void btn5_Click(object sender, EventArgs e)
    {
        if (btnEqualClicked)
        {
            numBuilder = "";
            btnEqualClicked = false;
        }
        numBuilder += "5";
        txtCalc.Text = numBuilder;
    }

    private void btn6_Click(object sender, EventArgs e)
    {
        if (btnEqualClicked)
        {
            numBuilder = "";
            btnEqualClicked = false;
        }
        numBuilder += "6";
        txtCalc.Text = numBuilder;
    }

    private void btn7_Click(object sender, EventArgs e)
    {
        if (btnEqualClicked)
        {
            numBuilder = "";
            btnEqualClicked = false;
        }
        numBuilder += "7";
        txtCalc.Text = numBuilder;
    }

    private void btn8_Click(object sender, EventArgs e)
    {
        if (btnEqualClicked)
        {
            numBuilder = "";
            btnEqualClicked = false;
        }
        numBuilder += "8";
        txtCalc.Text = numBuilder;
    }

    private void btn9_Click(object sender, EventArgs e)
    {
        if (btnEqualClicked)
        {
            numBuilder = "";
            btnEqualClicked = false;
        }
        numBuilder += "9";
        txtCalc.Text = numBuilder;
    }

    private void btn0_Click(object sender, EventArgs e)
    {
        if (btnEqualClicked)

```

```

    {
        numBuilder = "";
        btnEqualClicked = false;
    }
    numBuilder += "0";
    txtCalc.Text = numBuilder;
}

private void btnDecimal_Click(object sender, EventArgs e)
{
    // make sure that decimal is not clicked more than once
    if (!btnDecClicked)
    {
        if (btnEqualClicked)
        {
            numBuilder = "0";
            btnEqualClicked = false;
        }
        numBuilder += ".";
        txtCalc.Text = numBuilder;
        btnDecClicked = true;
    }
}

// operation buttons
private void btnAdd_Click(object sender, EventArgs e)
{
    try
    {
        // store the value in the number builder
        double num = Convert.ToDouble(numBuilder);

        // add the number to list
        listNumbers.Add(num);

        // reset the number builder
        numBuilder = "";

        // if equal sign is not clicked, perform operation on first and second item in number list
        // then reset the state of operator buttons
        if (!btnEqualClicked)
        {
            if (btnSubClicked)
            {
                calculate.Num1 = listNumbers[0];
                calculate.Num2 = listNumbers[1];
                res = calculate.Solve(listNumbers, "btnSubClicked");
                txtCalc.Text = res.ToString();

                toWrite.Output = $"{calculate.Num1} - {calculate.Num2} = {res}";
                toWrite.WriteCalcFile();

                btnSubClicked = false;
            }
            else if (btnMulClicked)
            {
                calculate.Num1 = listNumbers[0];
                calculate.Num2 = listNumbers[1];
                res = calculate.Solve(listNumbers, "btnMulClicked");
                txtCalc.Text = res.ToString();

                toWrite.Output = $"{calculate.Num1} * {calculate.Num2} = {res}";
                toWrite.WriteCalcFile();

                btnMulClicked = false;
            }
            else if (btnDivClicked)
            {
                calculate.Num1 = listNumbers[0];
                calculate.Num2 = listNumbers[1];
                res = calculate.Solve(listNumbers, "btnDivClicked");
                txtCalc.Text = res.ToString();
            }
        }
    }
}

```

```

        toWrite.Output = $"{calculate.Num1} / {calculate.Num2} = {res}";
        toWrite.WriteCalcFile();

        btnDivClicked = false;
    }
    else if (btnAddClicked)
    {
        calculate.Num1 = listNumbers[0];
        calculate.Num2 = listNumbers[1];
        res = calculate.Solve(listNumbers, "btnAddClicked");
        txtCalc.Text = res.ToString();

        toWrite.Output = $"{calculate.Num1} + {calculate.Num2} = {res}";
        toWrite.WriteCalcFile();

        btnAddClicked = false;
    }
}

// if equal sign is clicked
// from index 1, remove 2 items (values added by equal btn event)
// display the last value of result
else if (btnEqualClicked)
{
    listNumbers.RemoveRange(1, 2);
    txtCalc.Text = res.ToString();

    // reset the state of all btns
    btnEqualClicked = btnAddClicked = btnSubClicked = btnMulClicked = btnDivClicked = false;
}

// change state when btn is clicked
btnAddClicked = true;
}
catch (Exception)
{
    MessageBox.Show("Error! Please input a number.", "Invalid Input");
}
}

private void btnSub_Click(object sender, EventArgs e)
{
    try
    {
        double num = Convert.ToDouble(numBuilder);
        listNumbers.Add(num);
        numBuilder = "";

        if (!btnEqualClicked)
        {
            if (btnSubClicked)
            {
                calculate.Num1 = listNumbers[0];
                calculate.Num2 = listNumbers[1];
                res = calculate.Solve(listNumbers, "btnSubClicked");
                txtCalc.Text = res.ToString();

                toWrite.Output = $"{calculate.Num1} - {calculate.Num2} = {res}";
                toWrite.WriteCalcFile();

                btnSubClicked = false;
            }
            else if (btnMulClicked)
            {
                calculate.Num1 = listNumbers[0];
                calculate.Num2 = listNumbers[1];
                res = calculate.Solve(listNumbers, "btnMulClicked");
                txtCalc.Text = res.ToString();

                toWrite.Output = $"{calculate.Num1} * {calculate.Num2} = {res}";
                toWrite.WriteCalcFile();

                btnMulClicked = false;
            }

```

```

    }
    else if (btnDivClicked)
    {
        calculate.Num1 = listNumbers[0];
        calculate.Num2 = listNumbers[1];
        res = calculate.Solve(listNumbers, "btnDivClicked");
        txtCalc.Text = res.ToString();

        toWrite.Output = $"{calculate.Num1} / {calculate.Num2} = {res}";
        toWrite.WriteCalcFile();

        btnDivClicked = false;
    }
    else if (btnAddClicked)
    {
        calculate.Num1 = listNumbers[0];
        calculate.Num2 = listNumbers[1];
        res = calculate.Solve(listNumbers, "btnAddClicked");
        txtCalc.Text = res.ToString();

        toWrite.Output = $"{calculate.Num1} + {calculate.Num2} = {res}";
        toWrite.WriteCalcFile();

        btnAddClicked = false;
    }
    }
    else if (btnEqualClicked)
    {
        listNumbers.RemoveRange(1, 2);
        txtCalc.Text = res.ToString();
        btnEqualClicked = btnAddClicked = btnSubClicked = btnMulClicked = btnDivClicked = false;
    }
    }

    btnSubClicked = true;
}
catch (Exception)
{
    MessageBox.Show("Error! Please input a number.", "Invalid Input");
}
}

private void btnMul_Click(object sender, EventArgs e)
{
    try
    {
        double num = Convert.ToDouble(numBuilder);
        listNumbers.Add(num);
        numBuilder = "";

        if (!btnEqualClicked)
        {
            if (btnSubClicked)
            {
                calculate.Num1 = listNumbers[0];
                calculate.Num2 = listNumbers[1];
                res = calculate.Solve(listNumbers, "btnSubClicked");
                txtCalc.Text = res.ToString();

                toWrite.Output = $"{calculate.Num1} - {calculate.Num2} = {res}";
                toWrite.WriteCalcFile();

                btnSubClicked = false;
            }
            else if (btnMulClicked)
            {
                calculate.Num1 = listNumbers[0];
                calculate.Num2 = listNumbers[1];
                res = calculate.Solve(listNumbers, "btnMulClicked");
                txtCalc.Text = res.ToString();

                toWrite.Output = $"{calculate.Num1} * {calculate.Num2} = {res}";
                toWrite.WriteCalcFile();
            }
        }
    }
    catch (Exception)
    {
        MessageBox.Show("Error! Please input a number.", "Invalid Input");
    }
}

```

```

        btnMulClicked = false;
    }
    else if (btnDivClicked)
    {
        calculate.Num1 = listNumbers[0];
        calculate.Num2 = listNumbers[1];
        res = calculate.Solve(listNumbers, "btnDivClicked");
        txtCalc.Text = res.ToString();

        toWrite.Output = $"{calculate.Num1} / {calculate.Num2} = {res}";
        toWrite.WriteCalcFile();

        btnDivClicked = false;
    }
    else if (btnAddClicked)
    {
        calculate.Num1 = listNumbers[0];
        calculate.Num2 = listNumbers[1];
        res = calculate.Solve(listNumbers, "btnAddClicked");
        txtCalc.Text = res.ToString();

        toWrite.Output = $"{calculate.Num1} + {calculate.Num2} = {res}";
        toWrite.WriteCalcFile();

        btnAddClicked = false;
    }
}
else if (btnEqualClicked)
{
    listNumbers.RemoveRange(1, 2);
    txtCalc.Text = res.ToString();
    btnEqualClicked = btnAddClicked = btnSubClicked = btnMulClicked = btnDivClicked = false;
}

btnMulClicked = true;
}
catch (Exception)
{
    MessageBox.Show("Error! Please input a number.", "Invalid Input");
}
}

private void btnDiv_Click(object sender, EventArgs e)
{
    try
    {
        double num = Convert.ToDouble(numBuilder);
        listNumbers.Add(num);
        numBuilder = "";

        if (!btnEqualClicked)
        {
            if (btnSubClicked)
            {
                calculate.Num1 = listNumbers[0];
                calculate.Num2 = listNumbers[1];
                res = calculate.Solve(listNumbers, "btnSubClicked");
                txtCalc.Text = res.ToString();

                toWrite.Output = $"{calculate.Num1} - {calculate.Num2} = {res}";
                toWrite.WriteCalcFile();

                btnSubClicked = false;
            }
            else if (btnMulClicked)
            {
                calculate.Num1 = listNumbers[0];
                calculate.Num2 = listNumbers[1];
                res = calculate.Solve(listNumbers, "btnMulClicked");
                txtCalc.Text = res.ToString();

                toWrite.Output = $"{calculate.Num1} * {calculate.Num2} = {res}";
                toWrite.WriteCalcFile();
            }

```

```

        btnMulClicked = false;
    }
    else if (btnDivClicked)
    {
        calculate.Num1 = listNumbers[0];
        calculate.Num2 = listNumbers[1];
        res = calculate.Solve(listNumbers, "btnDivClicked");
        txtCalc.Text = res.ToString();

        toWrite.Output = $"{calculate.Num1} / {calculate.Num2} = {res}";
        toWrite.WriteCalcFile();

        btnDivClicked = false;
    }
    else if (btnAddClicked)
    {
        calculate.Num1 = listNumbers[0];
        calculate.Num2 = listNumbers[1];
        res = calculate.Solve(listNumbers, "btnAddClicked");
        txtCalc.Text = res.ToString();

        toWrite.Output = $"{calculate.Num1} + {calculate.Num2} = {res}";
        toWrite.WriteCalcFile();

        btnAddClicked = false;
    }
}
else if (btnEqualClicked)
{
    listNumbers.RemoveRange(1, 2);
    txtCalc.Text = res.ToString();
    btnEqualClicked = btnAddClicked = btnSubClicked = btnMulClicked = btnDivClicked = false;
}

btnDivClicked = true;
}
catch (Exception)
{
    MessageBox.Show("Error! Please input a number.", "Invalid Input");
}
}

private void btnEqual_Click(object sender, EventArgs e)
{
    try
    {
        double num = Convert.ToDouble(numBuilder);
        listNumbers.Add(num);

        // store a copy of the number in a temp variable
        double temp = num;

        if (btnAddClicked)
        {
            calculate.Num1 = listNumbers[0];
            calculate.Num2 = listNumbers[1];
            res = calculate.Add();

            txtCalc.Text = res.ToString();

            toWrite.Output = $"{calculate.Num1} + {calculate.Num2} = {res}";
            toWrite.WriteCalcFile();

            // clear the list
            // add the result and temp value into the list
            // res: index 0, temp: index 1

            listNumbers.Clear();
            listNumbers.Add(res);
            listNumbers.Add(temp);

            // this will allow user to perform last operation
            // using the last number every time equal btn is clicked
        }
    }
}

```

```

else if (btnSubClicked)
{
    calculate.Num1 = listNumbers[0];
    calculate.Num2 = listNumbers[1];
    res = calculate.Sub();

    txtCalc.Text = res.ToString();

    toWrite.Output = $"{calculate.Num1} - {calculate.Num2} = {res}";
    toWrite.WriteCalcFile();

    listNumbers.Clear();
    listNumbers.Add(res);
    listNumbers.Add(temp);
}
else if (btnMulClicked)
{
    calculate.Num1 = listNumbers[0];
    calculate.Num2 = listNumbers[1];
    res = calculate.Mul();

    txtCalc.Text = res.ToString();

    toWrite.Output = $"{calculate.Num1} * {calculate.Num2} = {res}";
    toWrite.WriteCalcFile();

    listNumbers.Clear();
    listNumbers.Add(res);
    listNumbers.Add(temp);
}
else if (btnDivClicked)
{
    calculate.Num1 = listNumbers[0];
    calculate.Num2 = listNumbers[1];
    res = calculate.Div();

    txtCalc.Text = res.ToString();

    toWrite.Output = $"{calculate.Num1} / {calculate.Num2} = {res}";
    toWrite.WriteCalcFile();

    listNumbers.Clear();
    listNumbers.Add(res);
    listNumbers.Add(temp);
}

// change state of equal btn
btnEqualClicked = true;
}
catch (Exception)
{
    MessageBox.Show("Error! Please input a number.", "Invalid Input");
}

}

private void btnCalcClear_Click(object sender, EventArgs e)
{
    numBuilder = "";
    txtCalc.Text = "";
    listNumbers.Clear();
    btnEqualClicked = btnAddClicked = btnSubClicked = btnMulClicked = btnDivClicked = false;
}

private void btnCalcExit_Click(object sender, EventArgs e)
{
    if (MessageBox.Show("Do you want to close this window? ", "Close Calculator", MessageBoxButtons.YesNo)
    == DialogResult.Yes)
    {
        this.Close();
    }
}
}
}

```



```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Text.RegularExpressions;
using System.Net;

namespace WinFormProject
{
    public partial class frmIPv4Validator : Form
    {
        public frmIPv4Validator()
        {
            InitializeComponent();

            DateTime openFormTime, closeFormTime;
            private void frmIPv4Validator_Load(object sender, EventArgs e)
            {
                openFormTime = DateTime.Now;
                lblIPValDate.Text = openFormTime.ToLongDateString();
            }

            private void btnIPVal_Click(object sender, EventArgs e)
            {
                // examples: 192.168.0.10 / 192.186.000
                string ipAddress = txtIPVal.Text.Trim();
                if (Regex.IsMatch(ipAddress, @"^(25[0-5]|2[0-4]\d|[0-1]?[0-9]\d?\d)(\. (25[0-5]|2[0-4]\d|[0-1]?[0-9]\d?\d)){3}$"))
                {
                    MessageBox.Show($"{ipAddress}\nThis IP is correct", "Valid IP");

                    DataStream toWrite = new DataStream();
                    toWrite.FileName = "BinaryIPv4";
                    toWrite.MsgBoxTitle = "IPv4 Validator";
                    toWrite.Output = ipAddress;

                    toWrite.WriteBinFile();
                } else if (ipAddress.Length == 0)
                {
                    MessageBox.Show("Error! Please input an IPv4 address.", "Invalid IP");
                } else
                {
                    MessageBox.Show($"{ipAddress}\n" +
                        $"The IP must have 4 bytes\n" +
                        $"Integer number between 0 to 255\n" +
                        $"Separated by a dot (255.255.255.255)", "Invalid IP");
                }
            }

            private void btnIPValReset_Click(object sender, EventArgs e)
            {
                txtIPVal.Text = "";
            }

            private void btnIPValExit_Click(object sender, EventArgs e)
            {
                closeFormTime = DateTime.Now;
                TimeSpan totRunTime = closeFormTime.Subtract(openFormTime);

                if (MessageBox.Show("Do you want to close this window?\n" +
                    $"Total runtime: {totRunTime.Minutes} mins and {totRunTime.Seconds} seconds", "Close IPv4 Validator", MessageBoxButtons.YesNo) == DialogResult.Yes)
                {
                    this.Close();
                }
            }
        }
    }
}

```

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

namespace WinFormProject
{
    public partial class frmReadFile : Form
    {
        public frmReadFile()
        {
            InitializeComponent();
        }

        public string fileOutput { get; set; }
        public string frmTitle { get; set; }

        private void frmReadFile_Load(object sender, EventArgs e)
        {
            rtfReadBox.Text = fileOutput;
            lblReadTitle.Text = frmTitle;
        }

        private void btnExit_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```

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

Class/Method Name	Description
1. <a href="#">DataStream.cs</a>	This class handles the writing and reading of both text and binary files of all forms
2. <code>public void CreateDir()</code>	Function to check if directory exists. If directory does not exist, create it.
3. <code>public void WriteFile()</code>	Function to write to text file
4. <code>public string ReadFile()</code>	Function to read from text file
5. <code>public void WriteCalcFile()</code>	Function to write the calculator file
6. <code>public void WriteBinFile()</code>	Function to write the binary file
7. <a href="#">CurrencyEx.cs</a>	This class handles the calculations for the currency conversions
8. <code>public decimal ConvertToCAD (string currencyFrom)</code>	Function to convert from any currency to Canadian Dollar (CAD)
9. <code>public decimal ConvertToCurrency(decimal convCAD, string currencyTo)</code>	Function to convert from Canadian Dollar (CAD) to any currency
10. <a href="#">ConvertTemp.cs</a>	This class handles the computations for temperature conversion and generate the temperature description
11. <code>public decimal ConvertToCelcius()</code>	Function to convert from Fahrenheit (F) to Celsius (C)
12. <code>public decimal ConvertToFahrenheit()</code>	Function to convert from Fahrenheit (F) to Celsius (C)
13. <code>public string DescribeTemperature(char unitFrom)</code>	Function to generate the temperature description for display
14. <a href="#">Calculator.cs</a>	This class handles all the computations for the operations and management of input list
15. <code>public double Add()</code>	Function to add two numbers
16. <code>public double Sub()</code>	Function to subtract two numbers
17. <code>public double Mul()</code>	Function to multiply two numbers
18. <code>public double Div()</code>	Function to divide two numbers
19. <code>public double Solve (List&lt;double&gt; listNumbers, string btn)</code>	Function to perform calculation based on activated operation and manage the list of numbers

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

For me, I took the most time building the calculator. The hardest part in doing it was making it work like an actual calculator. I started by creating a code that would handle operations for two numbers, but it got more complicated as I try to handle all possible scenarios like performing multiple operations on multiple numbers. However, the rest of the forms in the project went smoothly. I opted to create separate classes for the forms because I found it more convenient, and it made my code look cleaner and simpler to understand.