

PROJECT REPORT

OF

“ SCIENTIFIC CALCULATOR ”



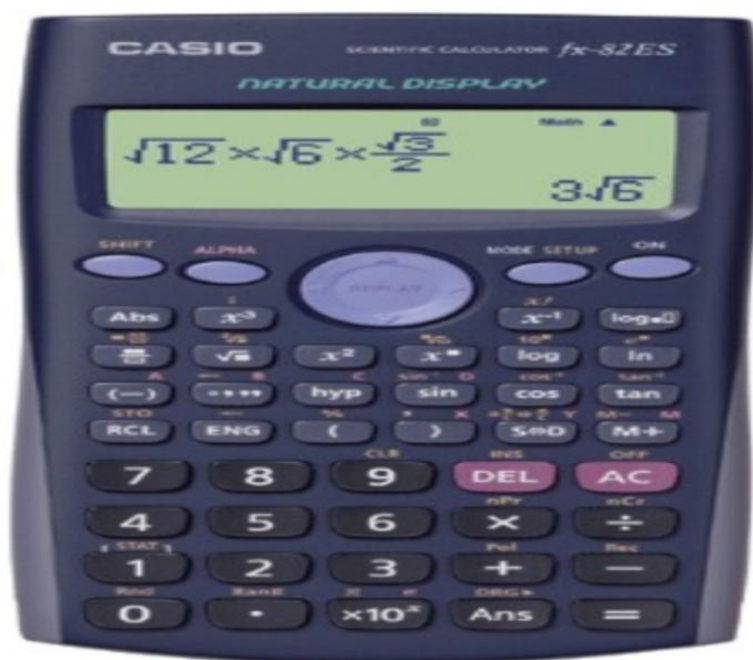
Delhi Technological University

Submitted in the partial fulfillment of the Degree of bachelor of Technology
In
Software Engineering

SUBMITTED BY :-
Shubham (2K19/SE/119)

GUIDED BY :-
Mr. Manoj Sethi

SCIENTIFIC CALCULATOR



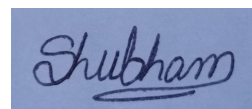
DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi - 110042

CANDIDATE'S DECLARATION

I, (Shubham (2K19/SE/119)) student of B.Tech (Software Engineering), hereby declare that the project dissertation titled, "Scientific Calculator" which is submitted by me to Department of Software Engineering, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of degree of Bachelor of Technology, is original and not copied from any source without citation. This work has not previously formed basis for award of any degree, diploma associateship, fellowship or any other similar title or recognition.



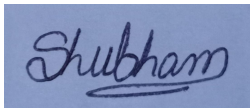
Date : November 18, 2020

Shubham (2K19/SE/119)

Place : Delhi Technological University, Delhi

ACKNOWLEDGEMENT

I, (Shubham (2K19/SE/119)) express my sincere gratitude to Mr. Manoj Sethi for his valuable guidance and timely suggestions during the entire duration of our dissertation work, without which this work would have been possible. I would like to convey my deep regards to all faculty members of Department of Software Engineering, who have bestowed their great efforts and guidance at appropriate time without which it would have been very difficult on our part to finish this work. I would like to thank my friends for their advice and pointing out the mistakes.

A blue rectangular box containing a handwritten signature in cursive script that reads "Shubham".

Shubham (2K19/SE/119)

ABSTRACT

This work was centered on the Design and implementation of a scientific calculator for education organization. The study traced calculator system as a tool to completely change mathematical knowledge and sophisticated problems solving strategies had advanced the field of simulated engine in mathematics.

Project is based on scientific calculator, along with which we have added some innovative components. There are eight component of this project are, standard calculator, scientific calculator, graphing calculator, quadratic equation solver, temperature converter, image viewer, notepad, paint. We have encapsulated all this feature into a single windows application.

This project work also focused principally on numbers and arithmetic operation. This researcher investigated the manual system in detail with a view to finding out the need to automate the system.

Interestingly, the end result of simple calculator system was its ability to process number and operators, and provides a useful result.

Therefore, this project will help immensely in the following way. Easy calculating of tedious mathematical problems, easy to retrieval of errors and it will also be of a good assistance to any researcher on these topics.

INDEX

Candidate's Declaration	ii
Acknowledgement	iii
Abstract	iv
Index	v
Introduction	1
Functions Provided	2
OOPs Concept Used	10
Whats Innovative	15
Conclusion	15
References	16
Annexure (Code)	17

INTRODUCTION

A scientific calculator is a type of electronic calculator, usually but not always handheld, designed to calculate problems in science (especially physics), engineering, and mathematics. They have almost completely replaced slide rules in almost all traditional applications, and are widely used in both education and professional settings.

A fully featured scientific calculator with proper operator precedence is implemented, including trig functions and logarithms, factorials, 12 levels of parentheses, logs to base 2 (a handy function for information entropists!), bitwise logical operator, hex, octal, binary and ASCII display.

The calculator is written in C# and you are welcome to view it for personal educational purposes as long as you recognize that it is copyrighted and not in the public domain. Project is based on scientific calculator, along with which we have added some innovative components. There are eight component of this project which are as follows :

1. Standard calculator
2. Scientific calculator
3. Graphing calculator
4. Quadratic equation solver
5. Temperature converter
6. Image viewer
7. Notepad
8. Paint

We have encapsulated all this feature into a single windows application.

FUNCTIONS PROVIDED

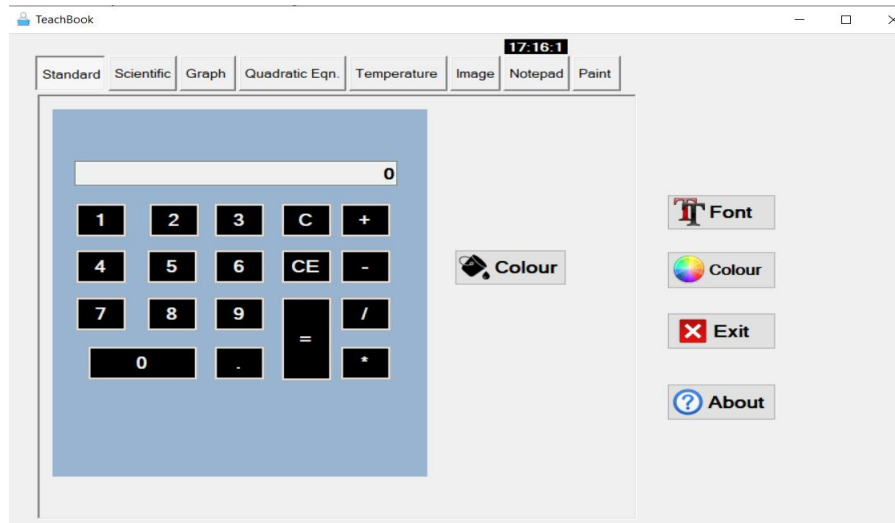
We have made a Windows application for this project. To open the application double click on the following item, then a new window will appear.



1. Standard Calculator

- Click on Standard tab to view the standard calculator.
- Numeric buttons from 1 to 9.
- C button for backspace and CE button to clear all text.
- Basic operation of +, -, *, /.
- A specific colour button to change the colour of the calculator.
- A specific text box showing current operation.

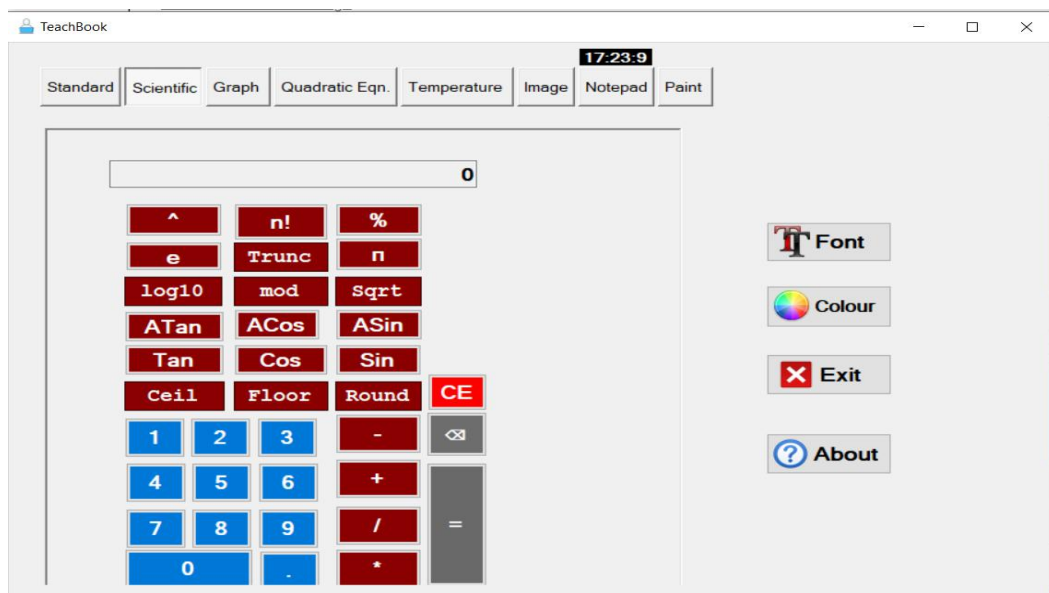
Standard calculator is used to perform basic arithmetic operations like +, -, *, /.



2. Scientific Calculator

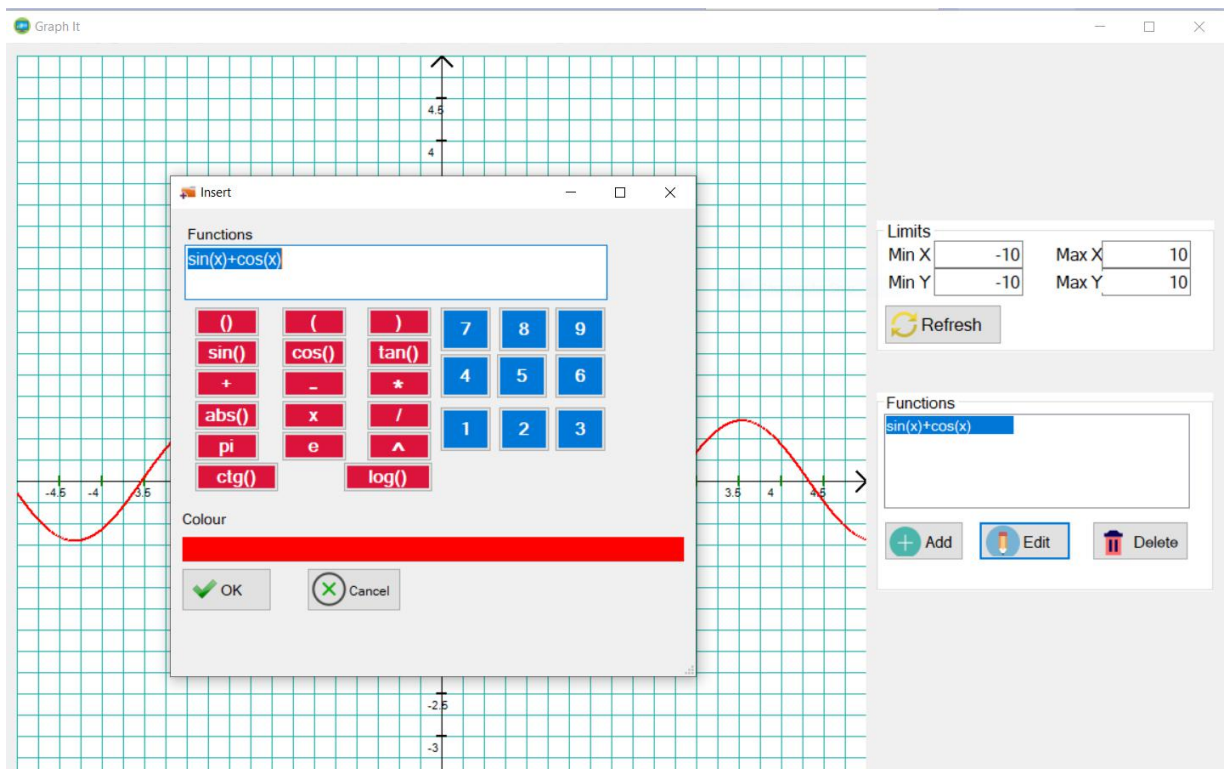
Click on Scientific tab to open scientific calculator.

- A fully functional standard calculator, with additional features like all trigonometric functions, inverse trigonometric functions, log function, factorial function, remainder operator, power function, exponential function and many more.
- We have added colourfull buttons to make it visually appealing.
- A small text box to view which operation has been performed.
- Just above the text box shows which functions had been performed out.



3. Graphing calculator

- Click on Graph tab and click on run graph button. A new window will appear running Graphing calculator.
- This tab deals with the drawing of graph related to maths function. The user needs to carefully write the equation and this tab will generate the graph.
- The user has to first set the limits of the graph i.e. x-axis limits and y-axis limits.
- Click on refresh button to update the graph.
- Click on add button for add new function and its graph.
- Now user has to carefully write the maths equations he/she wants to plot.
- Click on edit button for edit the existing function and its graph.
- Click on delete button for delete the existing function and its graph.
- User can change the colour of the graph according to his/her comfort.



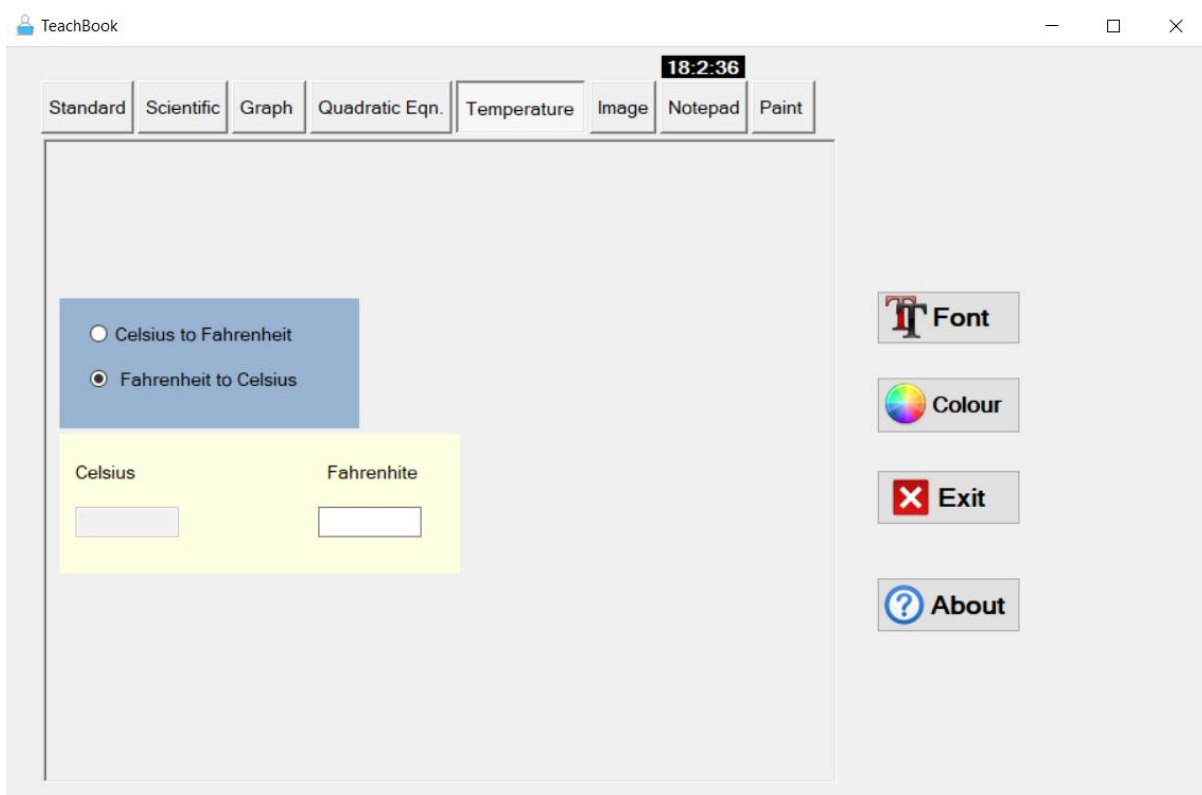
4. Quadratic Equation Solver

- Click on quadratic tab to open quadratic equation solver.
- Different position to enter different values i.e. different places to fill coefficient of x square, coefficient of x and constant.
- Click on solve button to solve the equation.
- Reset button to delete all entries.
- Our system is able to generate both types of roots, real roots as well as imaginary roots of the entered equation.
- Using exception handling it generate error message when user entered wrong input and when user doesn't give any input.

The screenshot shows a software window titled "TeachBook" with a standard Windows-style title bar (minimize, maximize, close buttons) and a digital clock displaying "17:52:23". The main interface features a horizontal menu bar with tabs: "Standard", "Scientific", "Graph", "Quadratic Eqn.", "Temperature", "Image", "Notepad", and "Paint". The "Quadratic Eqn." tab is currently selected. Below the menu bar is a large input area for the quadratic equation, structured as follows: a text box for the coefficient of x^2 , followed by a button labeled x^2 , a button labeled $+$, a text box for the coefficient of x , a button labeled x , a button labeled $+$, and a text box labeled "Constant". Below this input area, there are two rows for the roots: the first row has a button labeled x_1 , an equals sign button, and a text box; the second row has a button labeled x_2 , an equals sign button, and a text box. At the bottom of the input area are two buttons: "Reset" and "Solve". To the right of the main input area is a vertical sidebar containing four buttons: "Font" (with a text icon), "Colour" (with a color wheel icon), "Exit" (with a red X icon), and "About" (with a question mark icon).

5. Temperature Converter

- Click on Temperature tab to open temperature converter.
- It converts temperature from Fahrenheit to Celsius and vice - versa.
- Provides radio button for choose options.
- A special feature, when a user selects first option then Celsius button becomes active and Fahrenheit become inactive, so that user has to enter Celsius value to convert it to Fahrenheit. Reverse happens when user selects second option.



6. Image Viewer

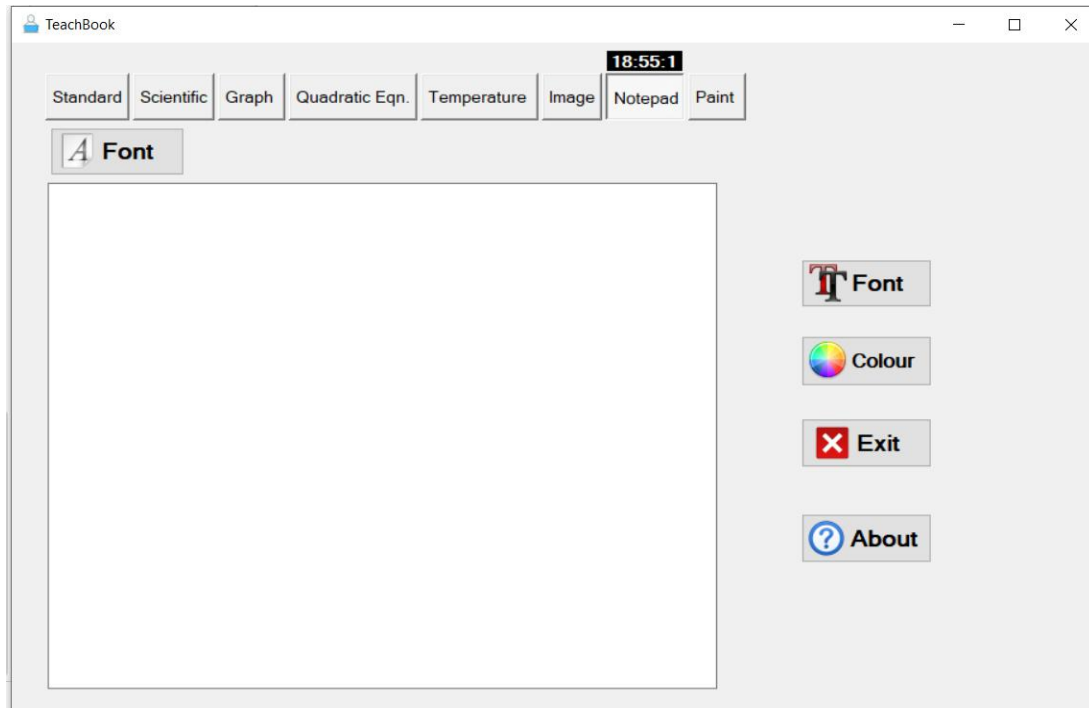
- Click on image tab to open image viewer.

- Click on browse button, a new window will appear.
- Browse through that new window and select your target image file.
- Click on open button the image will be displayed in the application.



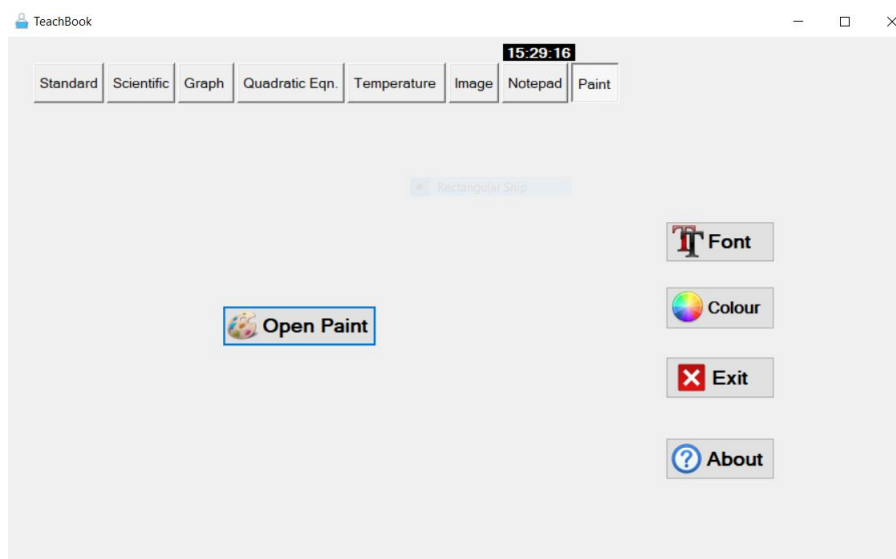
7. Notepad

- Click on Notepad to open it.
- The user can write anything he/she wants using the keyboard.
- Font button is provided, using this button the user can change the font type, font style, and other features related to font.



8. Paint

- Click on Paint to open it.
- The user can draw anything he/she wants using the functions provided in paint.

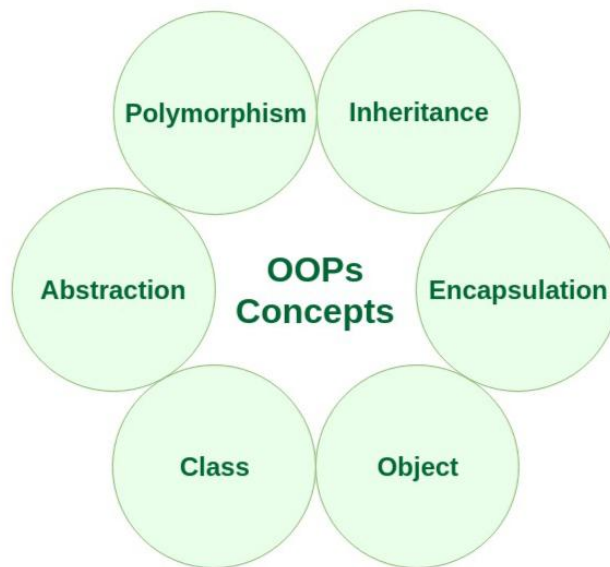


9. Some Miscellaneous Features Provided :

- Current time.
- Font button to modify the font of entire application.
- Colour button to change the background colour.
- Exit button to close the application.
- About button to show the information about the developer.



OOP CONCEPT USED



Class :

The building block of C++ that leads to Object-Oriented programming is a Class. It is a user-defined data type, which holds its own data members and member functions, which can be accessed and used by creating an instance of that class. A class is like a blueprint for an object.

A Class is a user-defined data-type which has data members and member functions.

Data members are the data variables and member functions are the functions used to manipulate these variables and together these data members and member functions define the properties and behaviour of the objects in a Class.

We can say that a Class in C++ is a blue-print representing a group of objects which shares some common properties and behaviours.

Object :

An Object is an identifiable entity with some characteristics and behaviour. An Object is an instance of a Class. When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated.

Object take up space in memory and have an associated address like a record in pascal or structure or union in C. When a program is executed the objects interact by sending messages to one another.

Each object contains data and code to manipulate the data. Objects can interact without having to know details of each other's data or code, it is sufficient to know the type of message accepted and type of response returned by the objects.

Encapsulation :

In normal terms, Encapsulation is defined as wrapping up of data and information under a single unit. In Object-Oriented Programming, Encapsulation is defined as binding together the data and the functions that manipulate them.

Consider a real-life example of encapsulation, in a company, there are different sections like the accounts section, finance section, sales section etc. The finance section handles all the financial transactions and keeps records of all the data related to finance. Similarly, the sales section

handles all the salesrelated activities and keeps records of all the sales. Now there may arise a situation when for some reason an official from the finance section needs all the data about sales in a particular month. In this case, he is not allowed to directly access the data of the sales section. He will first have to contact some other officer in the sales section and then request him to give the particular data. This is what encapsulation is. Here the data of the sales section and the employees that can manipulate them are wrapped under a single name “sales section”.

Encapsulation also leads to data abstraction or hiding. As using encapsulation also hides the data. In the above example, the data of any of the section like sales, finance or accounts are hidden from any other section.

Abstraction :

Data abstraction is one of the most essential and important features of objectoriented programming in C++. Abstraction means displaying only essential information and hiding the details. Data abstraction refers to providing only essential information about the data to the outside world, hiding the background details or implementation.

Consider a real-life example of a man driving a car. The man only knows that pressing the accelerators will increase the speed of the car or applying brakes will stop the car but he does not know about how on pressing accelerator the speed is actually increasing, he does not know about the inner mechanism of the car or the implementation of accelerator, brakes etc in the car. This is what abstraction is.

Abstraction using Classes: We can implement Abstraction in C++ using classes. The class helps us to group data members and member

functions using available access specifiers. A Class can decide which data member will be visible to the outside world and which is not.

Abstraction in Header files: One more type of abstraction in C++ can be header files. For example, consider the `pow()` method present in `math.h` header file. Whenever we need to calculate the power of a number, we simply call the function `pow()` present in the `math.h` header file and pass the numbers as arguments without knowing the underlying algorithm according to which the function is actually calculating the power of numbers.

Polymorphism :

The word polymorphism means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form.

A person at the same time can have different characteristics. Like a man at the same time is a father, a husband, an employee. So the same person possesses different behaviour in different situations. This is called polymorphism.

An operation may exhibit different behaviours in different instances. The behaviour depends upon the types of data used in the operation.

We use operator overloading and function overloading, C++ supports operator overloading and function overloading.

Operator Overloading :

The process of making an operator to exhibit different behaviours in different instances is known as operator overloading.

Function Overloading :

Function overloading is using a single function name to perform different types of tasks.

Polymorphism is extensively used in implementing inheritance.

Inheritance :

The capability of a class to derive properties and characteristics from another class is called Inheritance. Inheritance is one of the most important features of Object-Oriented Programming.

Sub Class : The class that inherits properties from another class is called Sub class or Derived Class.

Super Class : The class whose properties are inherited by sub class is called Base Class or Super class.

Reusability :

Inheritance supports the concept of “reusability”, i.e. when we want to create a new class and there is already a class that includes some of the code that we want, we can derive our new class from the existing class. By doing this, we are reusing the fields and methods of the existing class.

WHATS INNOVATIVE

Coronavirus epidemic has changed the way of teaching. All the classes are being held online. Student might be able to cope up with the changing way of teaching, but teachers are facing a lots of difficulty specially a school teacher who have limited understanding in computers and internet.

Our application focuses on helping these people specially upper primary and secondary level maths teachers. We will be showing the live demo on how application will help these teachers.

CONCLUSION

The performance of this application is very motivating. It is great tool for primary and secondary level maths teachers. It is very easy to use. This project has potential to be a industry level application.

With increasing technology, it is the need of the hour to incorporate this type of application in school curriculum. This application will make both teaching and learning of mathematics and a lot more fun and interactive.

REFERENCES

Research Papers -

- [1] Banerjee, Ishan & Nguyen, Bao & Garousi, Vahid & Memon, Atif. Graphical user interface (GUI) testing: Systematic mapping and repository. Information and Software Technology.1,Oct,. 2013.
- [2] Dourish, Paul. Algorithms and their others : Algorithmic culture in context. Big Data & Society. 24,Aug,.2016.

Websites -

- [1] “Exceptions - Try..Catch”,Oct. 23, 2020.[Online]. Available: https://www.w3schools.com/cs/cs_exceptions.asp. [Accessed: Oct. 23, 2020].
- [2] “How to open a new form from another form”,Nov.30,2020.[Online]. Available:<https://stackoverflow.com/questions/3965043/how-to-open-a-new-form-from-another-form>. [Accessed:Nov.30,2020].
- [3] “Working with Date and Time”,Dec.10,2020.[Online].Available:<https://www.tutorialsteacher.com/csharp/csharp-datetime>. [Accessed:Dec.5,2020].

ANNEXURE

Get our application on :

<https://drive.google.com/file/d/10jHBLB59dOVf62fnwV8XZo2esQ-BYqxy/view?usp=sharing>

CODE OF THE PROJECT

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
//namespace WindowsFormsApplication1
namespace WindowsFormsApp3
{
    public partial class Form1 : Form
    {
        double resultValue;
        String operation_performed="";
        bool eql = false;
        double resultValue2=0.0;
        String operation_performed2 = "";
        bool eql2 = false;
        double resultvalue3 = 0.0; double firstvalue, secondvalue;
        String operation_performed3= "";
        String x, s, xl, sl;
        int cel = 0,
            fr = 0;

        public Form1()
        {
```

```

        InitializeComponent();
    }

    private void btn_click(object sender, EventArgs e)
    {
        if ((textBox1.Text == "0"))
            textBox1.Clear();
        Button btn = (Button)sender;
        if (btn.Text == ".")
        {
            if (!textBox1.Text.Contains("."))
                textBox1.Text = textBox1.Text + btn.Text;
        }
        else
            textBox1.Text = textBox1.Text + btn.Text;
        eql = false;
    }

    private void operations(object sender, EventArgs e)
    {
        Button btn2 = (Button)sender;

        if ((resultValue != 0) && (eql == false))
        {
            //operation_performed = btn2.Text;
            button17.PerformClick();
            //resultValue = Double.Parse(textBox1.Text);

            textBox1.Clear();
            label1.Text = resultValue + " " + operation_performed;
        }
        else if ((resultValue != 0) && (eql == true))
        {
            textBox1.Text = resultValue.ToString();
            operation_performed = btn2.Text;
            //resultValue = Double.Parse(textBox1.Text);
            textBox1.Clear();
            label1.Text = resultValue + " " + operation_performed;
        }
        else
        {
            operation_performed = btn2.Text;
            resultValue = Double.Parse(textBox1.Text);
            textBox1.Clear();
            label1.Text = resultValue + " " + operation_performed;
        }
    }

```



```

    }

}

private void button4_Click(object sender, EventArgs e)
{
    textBox1.Text="0";

}

private void button7_Click(object sender, EventArgs e)
{
    textBox1.Text="0";
    resultValue = 0;
    label1.Text = "0";

}

private void button17_Click(object sender, EventArgs e)
{
    eql = true;
    switch (operation_performed)
    {
        case "+":
            label1.Text = label1.Text+" "+ textBox1.Text;
            //textBox1.Text = (resultValue + Double.Parse(textBox1.Text)).ToString();
            if (textBox1.Text == "")
            {
                textBox1.Text = (resultValue).ToString();
            }
            else
            {
                resultValue = resultValue + Double.Parse(textBox1.Text);
                textBox1.Text = (resultValue).ToString();
            }

            break;
        case "-":
            label1.Text = label1.Text + " " + textBox1.Text;
            //textBox1.Text = (resultValue + Double.Parse(textBox1.Text)).ToString();
            if (textBox1.Text == "")
            {
                textBox1.Text = (resultValue).ToString();
            }
            else
            {

```

```

        resultValue = resultValue - Double.Parse(textBox1.Text);
        textBox1.Text = (resultValue).ToString();
    }
    break;
case "/":
    label1.Text = label1.Text + " " + textBox1.Text;
    //textBox1.Text = (resultValue + Double.Parse(textBox1.Text)).ToString();
    if (textBox1.Text == "")
    {
        textBox1.Text = (resultValue).ToString();
    }
    else
    {
        resultValue = resultValue / Double.Parse(textBox1.Text);
        textBox1.Text = (resultValue).ToString();
    }
    break;
case "*":
    label1.Text = label1.Text + " " + textBox1.Text;
    //textBox1.Text = (resultValue + Double.Parse(textBox1.Text)).ToString();
    if (textBox1.Text == "")
    {
        textBox1.Text = (resultValue).ToString();
    }
    else
    {
        resultValue = resultValue * Double.Parse(textBox1.Text);
        textBox1.Text = (resultValue).ToString();
    }
    break;
default:
    break;
}
resultValue = Double.Parse(textBox1.Text);
label1.Text = "";

}

private void button19_Click(object sender, EventArgs e)
{
    MessageBox.Show("hi");
}

/// <summary>
/// ///tab 2 stats here
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void button21_Click(object sender, EventArgs e)
{
    if ((textBox2.Text == "0"))

```

```

        textBox2.Clear();
        Button btn = (Button)sender;
        if (btn.Text == ".")
        {
            if (!textBox2.Text.Contains("."))
                textBox2.Text = textBox2.Text + btn.Text;
        }
        else
            textBox2.Text = textBox2.Text + btn.Text;
        eql2 = false;
        label2.Text = label2.Text + " " + btn.Text;
    }

    private void button28_Click(object sender, EventArgs e)
    {

        Button btn2 = (Button)sender;
        label2.Text = label2.Text + btn2.Text;

        if ((resultValue2 != 0) && (eql2 == false))
        {

            operation_performed2 = btn2.Text;
            // button17.PerformClick();
            button25.PerformClick();
            resultValue2 = Double.Parse(textBox2.Text);

            textBox2.Clear();
            // label2.Text = resultValue2 + " " + operation_performed2;
            String c = btn2.Text;
            label2.Text = label2.Text + textBox2.Text;

        }
        else if ((resultValue2 != 0) && (eql2 == true))
        {
            textBox2.Text = resultValue2.ToString();
            operation_performed2 = btn2.Text;
            resultValue2 = Double.Parse(textBox2.Text);
            textBox2.Clear();
            // label2.Text = resultValue2 + " " + operation_performed2;
            label2.Text = label2.Text + textBox2.Text;

        }
        else
        {
            operation_performed2 = btn2.Text;
            resultValue2 = Double.Parse(textBox2.Text);
            textBox2.Clear();
            // label2.Text = resultValue2 + " " + operation_performed2;
            label2.Text = label2.Text + textBox2.Text;

        }
    }

```

```

}

private void button25_Click(object sender, EventArgs e)
{
    eql2 = true;
    switch (operation_performed2)
    {
        case "+":
            //label2.Text = label2.Text + " =" + textBox2.Text;
            //textBox1.Text = (resultValue + Double.Parse(textBox1.Text)).ToString();
            if (textBox2.Text == "")
            {
                textBox2.Text = (resultValue2).ToString();
            }
            else
            {
                resultValue2 = resultValue2 + Double.Parse(textBox2.Text);
                textBox2.Text = (resultValue2).ToString();
            }

            break;
        case "-":
            // label2.Text = label2.Text + " " + textBox2.Text;
            //textBox1.Text = (resultValue + Double.Parse(textBox1.Text)).ToString();
            if (textBox2.Text == "")
            {
                textBox2.Text = (resultValue2).ToString();
            }
            else
            {
                resultValue2 = resultValue2 - Double.Parse(textBox2.Text);
                textBox2.Text = (resultValue2).ToString();
            }
            break;
        case "/":
            //label2.Text = label2.Text + " " + textBox2.Text;
            //textBox1.Text = (resultValue + Double.Parse(textBox1.Text)).ToString();
            if (textBox2.Text == "")
            {
                textBox2.Text = (resultValue2).ToString();
            }
            else
            {
                resultValue2 = resultValue2 / Double.Parse(textBox2.Text);
                textBox2.Text = (resultValue2).ToString();
            }
            break;
        case "*":
            // label2.Text = label2.Text + " " + textBox2.Text;

```

```

        //textBox1.Text = (resultValue + Double.Parse(textBox1.Text)).ToString();
        if (textBox2.Text == "")
        {
            textBox2.Text = (resultValue2).ToString();
        }
        else
        {
            resultValue2 = resultValue2 * Double.Parse(textBox2.Text);
            textBox2.Text = (resultValue2).ToString();
        }
        break;
    default:
        label2.Text = label2.Text + " ";
        break;
    }
}

private void button26_Click(object sender, EventArgs e)
{
    textBox2.Text = "";
}

private void button36_Click(object sender, EventArgs e)
{
    textBox2.Text = "0";
    resultValue2 = 0;
    label2.Text = "";
}

private void button39_Click(object sender, EventArgs e)
{
}

private void button40_Click(object sender, EventArgs e)
{
    label2.Text = label2.Text + " " + "Sin(" + textBox2.Text + ")";

    double inpt = Double.Parse(textBox2.Text);
    double rslt = Math.Sin(inpt);
    textBox2.Text = rslt.ToString();

}

private void button41_Click(object sender, EventArgs e)
{
    label2.Text = "Cos(" + textBox2.Text + ")";

    double inpt = Double.Parse(textBox2.Text);
    double rslt = Math.Cos(inpt);

```

```

        textBox2.Text = rslt.ToString();

    }

    private void button45_Click(object sender, EventArgs e)
    {
        Button btn = (Button)sender;
        /* if (textBox3.Text == "0")
        {
            textBox3.Clear();
            textBox3.Text = textBox3.Text + btn.Text;
        }
        /* else
            textBox3.Text = textBox3.Text + btn.Text;
        // label3.Text = label3.Text + btn.Text;
        if (btn.Text == ".")
        {
            if (!textBox1.Text.Contains("."))
                textBox1.Text = textBox1.Text + btn.Text;
            label3.Text = label3.Text + btn.Text;
        }
        else
        {
            textBox1.Text = textBox1.Text + btn.Text;
        }
        */
        if ((textBox3.Text == "0"))
            textBox3.Clear();
        //Button btn = (Button)sender;
        if (btn.Text == ".")
        {
            if (!textBox3.Text.Contains("."))
                textBox3.Text = textBox3.Text + btn.Text;
            label3.Text = label3.Text + btn.Text;
        }
        else
        {
            textBox3.Text = textBox3.Text + btn.Text;
            label3.Text = label3.Text + btn.Text;
        }
    }

    private void bodmas_operations(object sender, EventArgs e)
    {
        Button btn = (Button)sender;

```

```

        if(resultvalue3!=0)
        {
            button49.PerformClick();
        }
        operation_performed3 = btn.Text;
        resultvalue3 = Double.Parse(textBox3.Text);
        label3.Text = label3.Text + btn.Text;
        textBox3.Clear();
    }

    private void button60_Click(object sender, EventArgs e)
    {
        textBox3.Text = "";
        resultvalue3 = 0;
        label3.Text = "";
    }

    private void button50_Click(object sender, EventArgs e)
    {
        s = textBox3.Text;
        int l = s.Length;
        for (int i = 0; i < l - 1; i++)
        {
            x += s[i];
        }
        textBox3.Text= x;
        x = "";

        sl = label3.Text;
        int l2 = sl.Length;
        for (int i = 0; i < l2 - 1; i++)
        {
            xl += sl[i];
        }
        label3.Text = xl;
        xl = "";
    }

    private void button65_Click(object sender, EventArgs e)
    {
        double pie = 3.1415926535897932384626433832795;
        Button btn = (Button)sender;
        if (textBox3.Text == "0")
        {
            textBox3.Clear();
            textBox3.Text = textBox3.Text + pie.ToString();
        }
    }

```

```

        else
            textBox3.Text = textBox3.Text + pie.ToString();
        label3.Text = label3.Text + "π";
    }

    private void button66_Click(object sender, EventArgs e)
    {
        double ee = 2.71828182845904523;
        Button btn = (Button)sender;
        if (textBox3.Text == "0")
        {
            textBox3.Clear();
            textBox3.Text = textBox3.Text + ee.ToString();
        }
        else
            textBox3.Text = textBox3.Text + ee.ToString();
        label3.Text = label3.Text + "e";
    }

    private void one_operator(object sender, EventArgs e)
    {
        Button btn = (Button)sender;
        String op = btn.Text;
        if (textBox3.Text != "")
        {
            double val = Double.Parse(textBox3.Text);
            switch (op)
            {
                case "Sin":
                    double rslt1 = Math.Sin(val);
                    textBox3.Text = rslt1.ToString();
                    label3.Text = label3.Text + "Sin(<-)";
                    break;
                case "Cos":
                    double rslt2 = Math.Cos(val);
                    textBox3.Text = rslt2.ToString();
                    label3.Text = label3.Text + "Cos(<-)";
                    break;
                case "Tan":
                    double rslt3 = Math.Tan(val);
                    textBox3.Text = rslt3.ToString();
                    label3.Text = label3.Text + "Tan(<-)";
                    break;
                case "n!":
                    int m = int.Parse(textBox3.Text);
                    int sum = 1;
                    for (int i = 1; i <= m; i++)
                    {

```



```

        sum = sum * i;
    }
    textBox3.Text = sum.ToString();
    label3.Text = label3.Text + " ";
    break;
case "log10":

    double l;
    l = Math.Log(val);
    textBox3.Text = Convert.ToString(l);
    label3.Text = label3.Text + "log(<-)";

    break;
case "Ceil":
    double l2;
    l2 = Math.Ceiling(val);
    textBox3.Text = Convert.ToString(l2);
    label3.Text = label3.Text + "Ceil(<-)";
    break;
case "mod":
    if(val<0)
    {
        val = val * (-1);
        textBox3.Text = val.ToString();
        label3.Text = label3.Text + "mod(<-)";
    }
    else
    {
        textBox3.Text = val.ToString();
        label3.Text = label3.Text + "mod(<-)";
    }
    break;
case "Sqrt":
    double l3 = Math.Sqrt(val);
    textBox3.Text = l3.ToString();
    label3.Text = label3.Text + "Sqrt(<-)";
    break;
case "Round":
    textBox3.Text = (Math.Round(val)).ToString();
    label3.Text = label3.Text + "Round(<-)";
    break;
case "Floor":
    textBox3.Text = (Math.Floor(val)).ToString();
    label3.Text = label3.Text + "Floor(<-)";
    break;
case "Trunc":
    textBox3.Text = (Math.Truncate(val)).ToString();
    label3.Text = label3.Text + "Trunc(<-)";
    break;
case "ASin":
    textBox3.Text = (Math.Asin(val)).ToString();

```

```

        label3.Text = label3.Text + "ASin(<-)";

        break;
    case "ACos":
        textBox3.Text = (Math.Acos(val)).ToString();
        label3.Text = label3.Text + "ACos(<-)";

        break;
    case "ATan":
        textBox3.Text = (Math.Atan(val)).ToString();
        label3.Text = label3.Text + "ACos(<-)";

        break;

    }

}
else
{
    MessageBox.Show("Enter some value ", "Warning");
}
}

private void textBox4_TextChanged(object sender, EventArgs e)
{
    //  textBox4.Text = DateTime.Now.Hour.ToString() + ":" +
    DateTime.Now.Minute.ToString() + ":" + DateTime.Now.Second.ToString();
}

private void textBox3_TextChanged(object sender, EventArgs e)
{
}

private void button86_Click(object sender, EventArgs e)
{
    textBox5.Text = "";
    textBox6.Text = "";
    textBox7.Text = "";
    textBox8.Text = "";
    textBox9.Text = "";

}

private void button87_Click(object sender, EventArgs e)
{

```

```

        if((textBox5.Text=="")|| (textBox6.Text == "")|| (textBox7.Text == ""))
        {
            MessageBox.Show("Invalid Input ", "Error", MessageBoxButtons.OK,
        MessageBoxIcon.Error);
            button86.PerformClick();

        }
        else
        {
            double a, b, c, D;
            a = Double.Parse(textBox5.Text);
            b = Double.Parse(textBox6.Text);
            c = Double.Parse(textBox7.Text);
            D = b * b - 4 * a * c;
            if(D>=0)
            {
                double r1, r2;
                r2 = (-b + Math.Sqrt(D)) / 2 * a;
                r1 = (-b - Math.Sqrt(D)) / 2 * a;
                textBox8.Text = r1.ToString();
                textBox9.Text = r2.ToString();
            }
            else
            {
                D = D * (-1);
                double val = Math.Sqrt(D);
                double b2 = -b;

                String x, y;
                x = "(" + b2 + "+" + "i" + D.ToString() + ")" + "/" + (2 * a).ToString();
                y = "(" + b2 + "-" + "i" + D.ToString() + ")" + "/" + (2 * a).ToString();
                textBox8.Text = x;
                textBox9.Text = y;
            }
        }
    }

    private void label5_Click(object sender, EventArgs e)
    {
        //label5.Text = DateTime.Now.ToString();
    }

    private void timer1_Tick(object sender, EventArgs e)
    {
        label5.Text = DateTime.Now.Hour.ToString() + ":" + DateTime.Now.Minute.ToString()
        + ":" + DateTime.Now.Second.ToString();
    }

    private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)

```

```

{
}

private void button73_Click(object sender, EventArgs e)
{
}

private void btnRound_Click(object sender, EventArgs e)
{
    try
    {
        if (textBox3.Text.Length != 0)
        {
            double l;
            l = Math.Round(Convert.ToDouble(textBox3.Text));
            textBox3.Text = Convert.ToString(l);
        }
        // sign_Indicator = 1;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK,
        MessageBoxIcon.Error);
    }
}

private void button75_Click(object sender, EventArgs e)
{
}

private void button49_Click(object sender, EventArgs e)
{
    switch(operation_performed3)
    {
        case "+":
            if(textBox3.Text=="")
            {
                MessageBox.Show("Invalid
Option","Warning",MessageBoxButtons.OK,MessageBoxIcon.Warning);
                secondvalue = 0;

                resultvalue3 = resultvalue3 + secondvalue;
                textBox3.Text = resultvalue3.ToString();
            }
            else
                secondvalue = Double.Parse(textBox3.Text);
    }
}

```

```

        resultvalue3 = resultvalue3 + secondvalue;
        textBox3.Text = resultvalue3.ToString();

        break;
    case "-":
        if (textBox3.Text == "")
        {
            MessageBox.Show("Invalid Option", "Warning",
                MessageBoxButtons.OK, MessageBoxIcon.Warning);
            secondvalue = 0;

            resultvalue3 = resultvalue3 - secondvalue;
            textBox3.Text = resultvalue3.ToString();

        }
        else
            secondvalue = Double.Parse(textBox3.Text);

        resultvalue3 = resultvalue3 - secondvalue;
        textBox3.Text = resultvalue3.ToString();

        break;

    case "/":
        if (textBox3.Text == "")
        {
            MessageBox.Show("Invalid Option", "Warning",
                MessageBoxButtons.OK, MessageBoxIcon.Warning);
            secondvalue = 1;

            resultvalue3 = resultvalue3 / secondvalue;
            textBox3.Text = resultvalue3.ToString();

        }
        else
            secondvalue = Double.Parse(textBox3.Text);

        resultvalue3 = resultvalue3 / secondvalue;
        textBox3.Text = resultvalue3.ToString();

        break;
    case "*":
        if (textBox3.Text == "")
        {
            MessageBox.Show("Invalid Option", "Warning",
                MessageBoxButtons.OK, MessageBoxIcon.Warning);
            secondvalue = 1;

            resultvalue3 = resultvalue3 * secondvalue;
            textBox3.Text = resultvalue3.ToString();

```

```

    }
    else
        secondvalue = Double.Parse(textBox3.Text);

    resultvalue3 = resultvalue3 * secondvalue;
    textBox3.Text = resultvalue3.ToString();

    break;
case "^":

    if (textBox3.Text == "")
    {
        MessageBox.Show("Invalid Option", "Warning",
        MessageBoxButtons.OK, MessageBoxIcon.Warning);
        secondvalue = 1;

        resultvalue3 = Math.Pow(resultvalue3, secondvalue);
        textBox3.Text = resultvalue3.ToString();

    }
    else
        secondvalue = Double.Parse(textBox3.Text);

    resultvalue3 = Math.Pow(resultvalue3, secondvalue);
    textBox3.Text = resultvalue3.ToString();
    break;
case "%":
    if (textBox3.Text == "")
    {
        MessageBox.Show("Invalid Option", "Warning",
        MessageBoxButtons.OK, MessageBoxIcon.Warning);
        secondvalue = 1;

        resultvalue3 = resultvalue3 % secondvalue;
        textBox3.Text = resultvalue3.ToString();

    }
    else
        secondvalue = Double.Parse(textBox3.Text);

    resultvalue3 = resultvalue3 % secondvalue;
    textBox3.Text = resultvalue3.ToString();
    break;

default:
    MessageBox.Show("Invalid Option", "Warning", MessageBoxButtons.OK,
    MessageBoxIcon.Warning);
    break;

```

```

    }
}

private void fontDialog1_Apply(object sender, EventArgs e)
{
}

private void button12_Click(object sender, EventArgs e)
{
    this.fontDialog1.ShowDialog();
    this.Font = this.fontDialog1.Font;
}

private void button88_Click(object sender, EventArgs e)
{
    this.colorDialog1.ShowDialog();
    this.BackColor = this.colorDialog1.Color;
}

private void button90_Click(object sender, EventArgs e)
{
    this.colorDialog1.ShowDialog();
    this.panel1.BackColor = this.colorDialog1.Color;
}

private void button89_Click(object sender, EventArgs e)
{
    MessageBox.Show("Have a Nice Day ", "Bye!!", MessageBoxButtons.OK,
    MessageBoxIcon.Exclamation);
    this.Close();
}

private void button82_Click(object sender, EventArgs e)
{
    this.openFileDialog1.ShowDialog();
    String fn = openFileDialog1.FileName;
    pictureBox1.Image = Image.FromFile(fn);
}

private void pictureBox1_Click(object sender, EventArgs e)
{
}

```

```

private void button91_Click(object sender, EventArgs e)
{
    this.fontDialog1.ShowDialog();
    textBox4.Font = this.fontDialog1.Font;
}

private void button92_Click(object sender, EventArgs e)
{
    this.process1.StartInfo.FileName = "C:\\Windows\\System32\\mspaint.exe ";
    this.process1.Start();
}

private void button74_Click(object sender, EventArgs e)
{
}

private void pictureBox2_Click(object sender, EventArgs e)
{
}

private void button93_Click(object sender, EventArgs e)
{
    this.process1.StartInfo.FileName = "C:\\Graph.exe";
    this.process1.Start();
}
Form2 secondForm = new Form2();
private void button94_Click(object sender, EventArgs e)
{
    secondForm.Show();
}

private void txtCelsious_TextChanged(object sender, EventArgs e)
{
    if (txtCelsious.Text != "" && rbnCF.Checked == true)
    {
        cel = int.Parse(txtCelsious.Text);
        fr = (cel * 9) / 5 + 32;

        txtFahrenheit.Text = fr.ToString();
    }
}

private void txtFahrenheit_TextChanged(object sender, EventArgs e)
{
    if (txtFahrenheit.Text != "" && rbnFC.Checked == true)
    {
        fr = int.Parse(txtFahrenheit.Text);

```



```

        cel = (fr - 32) * 5 / 9;

        txtCelsious.Text = cel.ToString();
    }
}

private void rbnFC_CheckedChanged(object sender, EventArgs e)
{
    if (rbnFC.Checked == true)
    {
        txtFahrenheit.Enabled = true;
        txtCelsious.Enabled = false;
    }
}

private void rbnCF_CheckedChanged(object sender, EventArgs e)
{
    if (rbnCF.Checked == true)
    {
        txtFahrenheit.Enabled = false;
        txtCelsious.Enabled = true;
    }
}

private void Form1_Load(object sender, EventArgs e)
{
    rbnCF.Checked = true;
}
}

```

PROGRAM.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApp3
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}

```

```
}  
}
```

FORM 2

```
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 WindowsFormsApp3
```

```
{  
    public partial class Form2 : Form  
    {  
        public Form2()  
        {  
            InitializeComponent();  
        }  
  
        private void Form2_Load(object sender, EventArgs e)  
        {  
  
        }  
  
        private void label1_Click(object sender, EventArgs e)  
        {  
  
        }  
    }  
}
```

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

```
namespace GraphOfFunction
```

```
{  
    public partial class FormFunction : Form
```

```

{
    private FunctionColor fc;
    public FunctionColor Fc
    {
        get
        {
            return fc;
        }
    }
    public FormFunction(FunctionColor fc)
    {
        InitializeComponent();
        this.fc = fc;

        textBoxFunction.Text = fc.Function;
        panelColor.BackColor = fc.Color;
    }

    private void panelColor_Click(object sender, EventArgs e)
    {
        DialogResult dr = colorDialogFunction.ShowDialog();
        panelColor.BackColor = colorDialogFunction.Color;
    }

    private void buttonOk_Click(object sender, EventArgs e)
    {
        fc.Function = textBoxFunction.Text;
        fc.Color = panelColor.BackColor;

        this.DialogResult = DialogResult.OK;
        this.Visible = false;
    }

    private void buttonCancel_Click(object sender, EventArgs e)
    {
        this.DialogResult = DialogResult.Cancel;
        this.Visible = false;
    }

    private void buttonText_Click(object sender, EventArgs e)
    {
        textBoxFunction.Text = textBoxFunction.Text.Insert(
            textBoxFunction.SelectionStart, ((Button)sender).Text);
    }

    private void panelColor_Paint(object sender, PaintEventArgs e)
    {
    }
}

```

```
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using GraphOfFunction.SyntaxNodes;

namespace GraphOfFunction
{
    class SyntaxTree
    {
        static List<string> listOperations = new List<string>(new string[]
        { "+", "-", "*", "/", "--", "^", "sin", "cos", "tan", "ctg", "log", "abs" });
        static List<string> listBinaryOperations = new List<string>(new string[]
        { "+", "-", "*", "/", "^" });
        static List<string> listUnaryOperations = new List<string>(new string[]
        { "--" });
        static List<string> listFunctions = new List<string>(new string[]
        { "sin", "cos", "tan", "ctg", "log", "abs" });

        private ISyntaxNode root;

        public SyntaxTree(string function)
        {
            List<ISyntaxNode> listNode = BuildListSyntaxNode(function);
            root = BuildSyntaxTree(listNode);
        }

        public ISyntaxNode ReadVar(string function, int first, out int index)
        {
            string var = "";
            for (int i = first;
                i < function.Length && (Char.IsNumber(function[i]) ||
function[i] == '.')) ; i++)
            {
                if (function[i] == '.') var += ',';
                else var += function[i];
            }
            index = first + var.Length - 1;
            return new Var(Convert.ToDouble(var));
        }

        public ISyntaxNode ReadOperation(string function, int first, out int
index)
```

```

    {
        index = first;
        if (function[first] == '+') return new Add();
        if (function[first] == '-') return new Sub();
        if (function[first] == '*') return new Mul();
        if (function[first] == '/') return new Div();
        if (function[first] == '^') return new Pow();
        return null;
    }

    public ISyntaxNode ReadFunction(string function, int first, out int
index)
    {
        index = first;
        if (function.Substring(first, 3) == "sin")
        {
            index = first + 2;
            return new Sin();
        }

        if (function.Substring(first, 3) == "cos")
        {
            index = first + 2;
            return new Cos();
        }

        if (function.Substring(first, 3) == "tan")
        {
            index = first + 2;
            return new Tan();
        }

        if (function.Substring(first, 3) == "ctg")
        {
            index = first + 2;
            return new Ctg();
        }

        if (function.Substring(first, 3) == "log")
        {
            index = first + 2;
            return new Log();
        }

        if (function.Substring(first, 3) == "abs")
        {
            index = first + 2;
            return new Abs();
        }
    }

```

```

        return null;
    }

    private int CompareOperations(ISyntaxNode a, ISyntaxNode b, int
aCountBracket, int bCountBracket)
    {
        if (aCountBracket < bCountBracket) return 1;
        if (aCountBracket > bCountBracket) return -1;
        if (listOperations.IndexOf(a.ToStringValue()) <
listOperations.IndexOf(b.ToStringValue())) return 1;
        if (listOperations.IndexOf(a.ToStringValue()) >
listOperations.IndexOf(b.ToStringValue())) return -1;
        return 0;
    }

    private ISyntaxNode FindMinPrioritiOperation(List<ISyntaxNode>
listNode, out int position)
    {
        ISyntaxNode min = null;
        int countMinOperationBrachet = 0;
        int countBracketOpening = 0;
        int countBracketClosing = 0;
        position = 0;
        for (int i = 0; i < listNode.Count; i++)
        {
            if (listNode[i].ToStringValue() == "(") countBracketOpening++;
            if (listNode[i].ToStringValue() == ")") countBracketClosing++;
            if (listOperations.IndexOf(listNode[i].ToStringValue()) != -1)
            {
                if (min == null)
                {
                    min = listNode[i];
                    position = i;
                    countMinOperationBrachet = countBracketOpening -
countBracketClosing;
                    continue;
                }

                if (CompareOperations(min, listNode[i],
countMinOperationBrachet, countBracketOpening - countBracketClosing) == -1)
                {
                    min = listNode[i];
                    position = i;
                    countMinOperationBrachet = countBracketOpening -
countBracketClosing;
                }
            }
        }
        return min;
    }
}

```

```

public void DeleteExcessiveBrackets(List<ISyntaxNode> listNode)
{
    if (listNode.Count == 1) return;
    int countMinBracket = listNode.Count;
    int countBracketOpening = 0;
    int countBracketClosing = 0;
    for (int i = 0; i < listNode.Count; i++)
    {
        if (listNode[i].ToStringValue() != "(" &&
listNode[i].ToStringValue() != ")")
        {
            countBracketOpening = 0;
            countBracketClosing = 0;
            for (int k = i + 1; k < listNode.Count; k++)
            {
                if (listNode[k].ToStringValue() == "(")
countBracketOpening++;
                if (listNode[k].ToStringValue() == ")")
countBracketClosing++;
            }
            int countBracket = Math.Abs(countBracketOpening -
countBracketClosing);
            if (countBracket < countMinBracket) countMinBracket =
countBracket;
        }
    }
    listNode.RemoveRange(0, countMinBracket);
    listNode.RemoveRange(listNode.Count - countMinBracket,
countMinBracket);
}

private ISyntaxNode BuildSyntaxTree(List<ISyntaxNode> listNode)
{
    DeleteExcessiveBrackets(listNode);
    if (listNode.Count == 1) return listNode[0];

    int position;
    ISyntaxNode min = FindMinPrioritiOperation(listNode, out
position);
    if (listBinaryOperations.IndexOf(min.ToStringValue()) != -1)
    {
        BinaryOperation operation = min as BinaryOperation;
        operation.SetA(BuildSyntaxTree(listNode.GetRange(0,
position)));
        operation.SetB(BuildSyntaxTree(listNode.GetRange(position +
1,listNode.Count - (position +1))));
    }
    if (listUnaryOperations.IndexOf(min.ToStringValue()) != -1)
    {

```

```

        UnaryOperation operation = min as UnaryOperation;
        operation.SetA(BuildSyntaxTree(listNode.GetRange(position + 1,
listNode.Count - (position + 1))));
    }
    if (listFunctions.IndexOf(min.ToStringValue()) != -1)
    {
        Function function = min as Function;
        function.SetX(BuildSyntaxTree(listNode.GetRange(position + 1,
listNode.Count - (position + 1))));
    }
    return min;
}

private List<ISyntaxNode> BuildListSyntaxNode(string function)
{
    List<ISyntaxNode> listNode = new List<ISyntaxNode>();
    for (int i = 0; i < function.Length; i++)
    {
        if (function[i] == 'p')
        {
            ISyntaxNode pi = new Var(Math.PI);
            listNode.Add(pi);
            i++;
        }

        if (function[i] == 'e')
        {
            ISyntaxNode e = new Var(Math.E);
            listNode.Add(e);
        }

        if (function[i] == 'x')
        {
            ISyntaxNode x = new X();
            listNode.Add(x);
        }
        if (Char.IsNumber(function[i]))
        {
            ISyntaxNode var = ReadVar(function, i, out i);
            listNode.Add(var);
        }
        if (function[i] == '-')
        {
            if (i == 0 || function[i - 1] == '(' || function[i + 1] ==
'(')
            {
                ISyntaxNode op = new Neg();
                listNode.Add(op);
                continue;
            }
        }
    }
}

```



```

    }
    if (MyChar.IsBinaryOperation(function[i]))
    {
        ISyntaxNode op = ReadOperation(function, i, out i);
        listNode.Add(op);
    }
    if (function[i] == '(')
    {
        ISyntaxNode op = new BracketOpening();
        listNode.Add(op);
    }
    if (function[i] == ')')
    {
        ISyntaxNode op = new BracketClosing();
        listNode.Add(op);
    }
    if (function[i] == 's' || function[i] == 'c' || function[i] ==
't' || function[i] == 'l' || function[i] == 'a')
    {
        ISyntaxNode func = ReadFunction(function, i, out i);
        listNode.Add(func);
    }
}
return listNode;
}

public double Calculate(double x, out bool isDomainOfFunction)
{
    if (root.IsDomainOfFunction(x))
    {
        isDomainOfFunction = true;
        return root.GetResult(x);
    }
    isDomainOfFunction = false;
    return 0;
}

}

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.Speech.Synthesis;

```

```

using System.Speech.Recognition;

namespace GraphOfFunction
{
    public partial class FormMy : Form
    {
        public FormMy()
        {
            InitializeComponent();
        }

        private void DrawBackground()
        {
            int sizeX = panelGraphFunction.Size.Width;
            int sizeY = panelGraphFunction.Size.Height;
            double minX = -10, maxX = 10;
            double minY = -10, maxY = 10;

            minX = Convert.ToDouble(textBoxMinX.Text);
            maxX = Convert.ToDouble(textBoxMaxX.Text);
            minY = Convert.ToDouble(textBoxMinY.Text);
            maxY = Convert.ToDouble(textBoxMaxY.Text);

            Graphics g = panelGraphFunction.CreateGraphics();
            g.Clear(Color.White);

            for (int i = 0; i < sizeX; i += 20)
            {
                g.DrawLine(new Pen(Color.LightSeaGreen), new Point(i, 0), new
Point(i, sizeY));
            }
            for (int i = 0; i < sizeY; i += 20)
            {
                g.DrawLine(new Pen(Color.LightSeaGreen), new Point(0, i), new
Point(sizeX, i));
            }
            g.DrawLine(new Pen(Color.Black), new Point(0, sizeY / 2), new
Point(sizeX, sizeY / 2));
            g.DrawLine(new Pen(Color.Black, 2), new Point(sizeX - 10, sizeY /
2 - 10), new Point(sizeX, sizeY / 2));
            g.DrawLine(new Pen(Color.Black, 2), new Point(sizeX - 10, sizeY /
2 + 10), new Point(sizeX, sizeY / 2));
            g.DrawLine(new Pen(Color.Black), new Point(sizeX / 2, 0), new
Point(sizeX / 2, sizeY));
            g.DrawLine(new Pen(Color.Black, 2), new Point(sizeX / 2 - 10, 0 +
10), new Point(sizeX / 2, 0));
            g.DrawLine(new Pen(Color.Black, 2), new Point(sizeX / 2 + 10, 0 +
10), new Point(sizeX / 2, 0));
        }
    }
}

```

```

        for (int i = 40; i < sizeX; i += 40)
        {
            if (i == sizeX / 2) continue;
            string st = (minX + i * ((maxX - minX) / sizeX)).ToString();
            g.DrawLine(new Pen(Color.Green, 2), new Point(i, sizeY / 2 + 5),
new Point(i, sizeY / 2 - 5));
            g.DrawString(st, new Font("Arial", 8), new
SolidBrush(Color.Black), new PointF(i - 15, sizeY / 2 + 4));
        }
        for (int i = 40; i < sizeY; i += 40)
        {
            if (i == sizeY / 2) continue;
            string st = (minX + (sizeY - i) * ((maxY - minY) /
sizeY)).ToString();
            g.DrawLine(new Pen(Color.Black, 2), new Point(sizeX / 2 + 5, i),
new Point(sizeX / 2 - 5, i));
            g.DrawString(st, new Font("Arial", 8), new
SolidBrush(Color.Black), new PointF(sizeX / 2 - 15, i + 4));
        }
        string point = "(" + (minX + (maxX - minX) / 2) + "; " + (minX + (maxY
- minY) / 2) + ")";
        g.DrawString(point, new Font("Arial", 8), new
SolidBrush(Color.Black), new PointF(sizeX / 2 - 15, sizeY / 2 + 4));
    }

    private void DrawFunction(string function, Color functionColor)
    {
        int sizeX = panelGraphFunction.Size.Width;
        int sizeY = panelGraphFunction.Size.Height;
        double minX = -10, maxX = 10;
        double minY = -10, maxY = 10;

        minX = Convert.ToDouble(textBoxMinX.Text);
        maxX = Convert.ToDouble(textBoxMaxX.Text);
        minY = Convert.ToDouble(textBoxMinY.Text);
        maxY = Convert.ToDouble(textBoxMaxY.Text);

        SyntaxTree calculator = new SyntaxTree(function);

        Graphics g = panelGraphFunction.CreateGraphics();

        Point currentPoint = new Point();
        Point previousPoint = new Point();
        bool isCurentPoint = false;
        bool isPreviousPoint = false;
        for (double x = minX; x < maxX; x += (maxX - minX) / sizeX)
        {
            double y = calculator.Calculate(x, out isCurentPoint);
            if (Math.Abs(y) > maxY * 2) isCurentPoint = false;
            if (isCurentPoint)

```

```

        {
            currentPoint.X = (int)(x / ((maxX - minX) / sizeX) - minX /
((maxX - minX) / (double)sizeX));
            currentPoint.Y = (int)(sizeY - (y / ((maxY - minY) / sizeY)
- minY / ((maxY - minY) / (double)sizeY)));
            if (isPreviousPoint)
            {
                g.DrawLine(new Pen(functionColor, 2), previousPoint,
currentPoint);
            }
        }
        previousPoint = currentPoint;
        isPreviousPoint = isCurentPoint;
    }
}

private void buttonAdd_Click(object sender, EventArgs e)
{
    FunctionColor fc = new FunctionColor("", Color.Black);
    FormFunction formFunction = new FormFunction(fc);
    if (formFunction.ShowDialog() == DialogResult.OK)
    {
        listBoxFunctions.Items.Add(fc);
        DrawFunction(fc.Function, fc.Color);
    }
}

private void buttonEdit_Click(object sender, EventArgs e)
{
    if (listBoxFunctions.SelectedItem == null) return;
    FormFunction formFunction = new
FormFunction(listBoxFunctions.SelectedItem as FunctionColor);

    if (formFunction.ShowDialog() == DialogResult.OK)
    {
        listBoxFunctions.Items.Remove(listBoxFunctions.SelectedItem);
        listBoxFunctions.Items.Add(formFunction.Fc);

        DrawBackground();
        for (int i = 0; i < listBoxFunctions.Items.Count; i++)
        {
            FunctionColor fc = listBoxFunctions.Items[i] as
FunctionColor;
            DrawFunction(fc.Function, fc.Color);
        }
    }
}

private void buttonDelete_Click(object sender, EventArgs e)

```

```

    {
        if (listBoxFunctions.SelectedItem == null) return;

        listBoxFunctions.Items.Remove(listBoxFunctions.SelectedItem);

        DrawBackground();
        for (int i = 0; i < listBoxFunctions.Items.Count; i++)
        {
            FunctionColor fc = listBoxFunctions.Items[i] as FunctionColor;
            DrawFunction(fc.Function, fc.Color);
        }
    }

    private void panelGraphFunction_Paint(object sender, PaintEventArgs e)
    {
        DrawBackground();
    }

    private void buttonRefresh_Click(object sender, EventArgs e)
    {
        DrawBackground();
        for(int i = 0; i < listBoxFunctions.Items.Count; i++)
        {
            FunctionColor fc = listBoxFunctions.Items[i] as FunctionColor;
            DrawFunction(fc.Function, fc.Color);
        }
    }

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

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

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace GraphOfFunction
{

```

```

class MyChar
{
    public static bool IsBinaryOperation(char c)
    {
        if(c == '+' || c == '-' || c == '*' || c == '/' || c == '^') return
true;
        return false;
    }
}

```

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```

namespace GraphOfFunction.SyntaxNodes
{
    class Add: BinaryOperation
    {
        public Add()
        {
        }

        public Add(ISyntaxNode a, ISyntaxNode b)
            : base(a, b)
        {
        }

        public override string ToStringValue()
        {
            return "+";
        }

        public override double GetResult(double x)
        {
            return GetA().GetResult(x) + GetB().GetResult(x);
        }

        public override bool IsDomainOfFunction(double x)
        {
            return GetA().IsDomainOfFunction(x) && GetB().IsDomainOfFunction(x);
        }
    }
}

```

