```
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.IO;

namespace Graph_Plotter
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        /*
        x = -10001      - = -10005      . = -9999
        ( = -10002      * = -10006      y = -9997
        ) = -10003      / = -10007
        + = -10004      ^ = -9998
        sin = -10008    log    = -10011
        cos = -10009    ln     = -10012
        tan = -10010    antilog = -10013
        antiln = -10014 root = -10015

        asin = -10016   acos = -10017
        atan = -10018   floor = -10019
        abs = -10020    ceil = -10021
        frac = -10022
        int inToP(stack input)
        float pToIn(stack post, float radian)
        void DrawGraph(stack post)

        a = arbitrary constant

        int fType case:
            0 == normal;
            1 == parametric;
            2 == polar;
        */
        Pen FXPen = new Pen(Color.Cyan, 3);
        Pen AreaPen = new Pen(Color.FromArgb(0,40,170), 3);
        SolidBrush YYPen = new SolidBrush(Color.LightGreen);
        Pen GrayPen = new Pen(Color.Gray, 1);
        Pen axisPen = new Pen(Color.White, 2);
        SolidBrush FXbrush = new SolidBrush(Color.Cyan);
        SolidBrush GXbrush = new SolidBrush(Color.Orange);
        SolidBrush HXbrush = new SolidBrush(Color.Yellow);
        SolidBrush brush = new SolidBrush(Color.White);

        float xDown, yDown, xMove, yMove, xShift = 0, yShift = 0,a=1; //xUp, yUp;
        int fType = 0;
        string[] fnHistory = { "","","","","","","","","","","","","","",""};
        string path = @"D:\history.gpx";
        class stack
        {
            public
            float[] a = new float[50];
            public int top;
            public stack()
            {
                top = -1;
                for(int i=0;i<50;i++)
                {
                    a[i] = 0;
                }
            }
            public void push(float x)
            {
                top++;
                a[top] = x;
            }
            public float pop()
            {
                float x= a[top];
                top--;
                return x;
            }

        }
        stack FXinput = new stack();
        stack FXpost = new stack();
        stack GXinput = new stack();
        stack GXpost = new stack();
        stack HXinput = new stack();
        stack HXpost = new stack();
        int inToP(stack input,stack post)
        {
            stack temp = new stack();
            for(int i=0;i<=input.top;i++)
            {
                if (input.a[i] == -10002)// (
                    temp.push(-10002);
                else if (input.a[i] == -10004 || input.a[i] == -10005)// + or -
                {
                    while (temp.a[temp.top] != -10002)
                    {
                        post.push(temp.a[temp.top]);
                        temp.top--;
                    }
                    temp.push(input.a[i]);
                }
                else if (input.a[i] == -10003)// )
                {
                    while (temp.a[temp.top] != -10002)
                    {
                        post.push(temp.a[temp.top]);
                        temp.top--;
                    }
                    temp.top--;
                }
                else if (input.a[i] == -10001)// x
                    post.push(-10001);
                else if (input.a[i] == -9997)// y
                    post.push(-9997);
                else if (input.a[i] == -10006 || input.a[i] == -10007)// * or /
                {
                    while (temp.a[temp.top] != -10002 && temp.a[temp.top] != -10004
                      && temp.a[temp.top] != -10005)
                    {
                        post.push(temp.a[temp.top]);
                        temp.top--;
                    }
                    temp.push(input.a[i]);
                }
                else if (input.a[i] == -9998)// ^
                {
                    while (temp.a[temp.top] != -10002 && temp.a[temp.top] != -10004
                      && temp.a[temp.top] != -10005 && temp.a[temp.top] != -10006
                     && temp.a[temp.top] != -10007)
                    {
                        post.push(temp.a[temp.top]);
                        temp.top--;
                    }
                    temp.push(input.a[i]);
                }
                else if (input.a[i] <= -10008 && input.a[i] >= -10022)// sin/cos/
                  tan/log/ln/antilog/antiln/root/floor/ceil/frac/asin/acos/atan/abs
                {
                    while (temp.a[temp.top] != -10002 && temp.a[temp.top] != -10004
                      && temp.a[temp.top] != -10005 && temp.a[temp.top] != -10006
                     && temp.a[temp.top] != -10007 && temp.a[temp.top] != -9998)
                    {
                        post.push(temp.a[temp.top]);
                        temp.top--;
                    }
                    temp.push(input.a[i]);
                }
                else// constants
                {
                    int p, l, t = i, pointLoc = -1;
                    float s = 0;
                    for (l = 0; input.a[i] >= 0 || input.a[i] == -9999; l++, i++)
                    {
                        if (input.a[i] == -9999)
                            pointLoc = l;
                    }
                    if (pointLoc != -1)
                    {
                        int maxi = i - 1;
                        i = t;
                        for (p = pointLoc - 1; p >= 0; p--, i++)
                            s += input.a[i] * (float)Math.Pow(10, p);
                        i++;
                        for (; i <= maxi; i++, p--)
                            s += input.a[i] * (float)Math.Pow(10, p);
                    }
                    else
                    {
                        i = t;
                        for (p = l - 1; p >= 0; p--, i++)
                            s += input.a[i] * (float)Math.Pow(10, p);
                    }
                    post.push(s);
                    i--;
                }
            }
            if (temp.top == -1)
                return -1;
            else
                return 0;
        }

        float pToIn(stack post, float r/* r= radian*/)
        {
            stack temp = new stack();
            stack final = new stack();
            float b1, b2, b3;
            temp = post;
            int i;
            for(i=0;i<=temp.top;i++)
            {
                if (temp.a[i] == -10004)// +
                {
                    b1 = final.pop();
                    b2 = final.pop();
                    b3 = b1 + b2;
                    final.push(b3);
                }
                else if (temp.a[i] == -10001)// x
                    final.push(r);
                else if (temp.a[i] == -10005)// -
                {
                    b1 = final.pop();
                    b2 = final.pop();
                    b3 = b2 - b1;
                    final.push(b3);
                }
                else if (temp.a[i] == -10006)// *
                {
                    b1 = final.pop();
                    b2 = final.pop();
                    b3 = b1 * b2;
                    final.push(b3);
                }
                else if (temp.a[i] == -10007)// /
                {
                    b1 = final.pop();
                    b2 = final.pop();
```

```
                        b3 = b2 / b1;
                        final.push(b3);
                    }
                    else if (temp.a[i] == -9998)// ^
                    {
                        b1 = final.pop();
                        b2 = final.pop();
                        if (b2 < 0 && b1 < 1 && b1 > -1)
                            b3 = 0;
                        else
                            b3 = (float)Math.Pow(b2, b1);
                        final.push(b3);
                    }
                    else if (temp.a[i] == -10008)// sin
                    {
                        b1 = final.pop();
                        b2 = (float)Math.Sin(b1);
                        final.push(b2);
                    }
                    else if (temp.a[i] == -10020)// abs
                    {
                        b1 = final.pop();
                        if (b1 < 0)
                            b1 = -b1;
                        final.push(b1);
                    }
                    else if (temp.a[i] == -10019)// floor
                    {
                        b1 = final.pop();
                        b2 = (float)Math.Floor(b1);
                        final.push(b2);
                    }
                    else if (temp.a[i] == -10021)// ceiling
                    {
                        b1 = final.pop();
                        b2 = (float)Math.Ceiling(b1);
                        final.push(b2);
                    }
                    else if (temp.a[i] == -10022)// fraction
                    {
                        b1 = final.pop();
                        b2 = (float)(b1-Math.Floor(b1));
                        final.push(b2);
                    }
                    else if (temp.a[i] == -10016)// asin
                    {
                        b1 = final.pop();
                        b2 = (float)Math.Asin(b1);
                        final.push(b2);
                    }
                    else if (temp.a[i] == -10017)// acos
                    {
                        b1 = final.pop();
                        b2 = (float)Math.Acos(b1);
                        final.push(b2);
                    }
```

```
                else if (temp.a[i] == -10018)// atan
                {
                    b1 = final.pop();
                    b2 = (float)Math.Atan(b1);
                    final.push(b2);
                }
                else if (temp.a[i] == -10009)// cos
                {
                    b1 = final.pop();
                    b2 = (float)Math.Cos(b1);
                    final.push(b2);
                }
                else if (temp.a[i] == -10010)// tan
                {
                    b1 = final.pop();
                    b2 = (float)Math.Tan(b1);
                    final.push(b2);
                }
                else if (temp.a[i] == -10011)// log
                {
                    b1 = final.pop();
                    if (b1 <= 0)
                        b2 = -10000;
                    else
                        b2 = (float)Math.Log10(b1);
                    final.push(b2);
                }
                else if (temp.a[i] == -10012)// ln
                {
                    b1 = final.pop();
                    if (b1 <= 0)
                        b2 = -10000;
                    else
                        b2 = (float)Math.Log10(b1);
                    final.push(b2);
                }
                else if (temp.a[i] == -10013)// antilog
                {
                    b1 = final.pop();
                    b2 = (float)Math.Pow(10, b1);
                    final.push(b2);
                }
                else if (temp.a[i] == -10014)// antiln
                {
                    b1 = final.pop();
                    b2 = (float)Math.Pow(2.17, b1);
                    final.push(b2);
                }
                else if (temp.a[i] == -10015)// root
                {
                    b1 = final.pop();
                    b2 = (float)Math.Sqrt(b1);
                    final.push(b2);
                }
                else
                    final.push(temp.a[i]);
```

```
            }
            return final.a[0];
        }
        float pToIn(stack post, float r1/* r1= radian1*/, float r2/*r2=radian2*/)
        {
            stack temp = new stack();
            stack final = new stack();
            float b1, b2, b3;
            temp = post;
            int i;
            for (i = 0; i <= temp.top; i++)
            {
                if (temp.a[i] == -10004)// +
                {
                    b1 = final.pop();
                    b2 = final.pop();
                    b3 = b1 + b2;
                    final.push(b3);
                }
                else if (temp.a[i] == -10001)// x
                    final.push(r1);
                else if (temp.a[i] == -9997)// y
                    final.push(r2);
                else if (temp.a[i] == -10005)// -
                {
                    b1 = final.pop();
                    b2 = final.pop();
                    b3 = b2 - b1;
                    final.push(b3);
                }
                else if (temp.a[i] == -10006)// *
                {
                    b1 = final.pop();
                    b2 = final.pop();
                    b3 = b1 * b2;
                    final.push(b3);
                }
                else if (temp.a[i] == -10007)// /
                {
                    b1 = final.pop();
                    b2 = final.pop();
                    b3 = b2 / b1;
                    final.push(b3);
                }
                else if (temp.a[i] == -9998)// ^
                {
                    b1 = final.pop();
                    b2 = final.pop();
                    if (b2 < 0 && b1 < 1 && b1 > -1)
                        b3 = 0;
                    else
                        b3 = (float)Math.Pow(b2, b1);
                    final.push(b3);
                }
                else if (temp.a[i] == -10008)// sin
                {
                    b1 = final.pop();
                    b2 = (float)Math.Sin(b1);
                    final.push(b2);
                }
                else if (temp.a[i] == -10020)// abs
                {
                    b1 = final.pop();
                    if (b1 < 0)
                        b1 = -b1;
                    final.push(b1);
                }
                else if (temp.a[i] == -10019)// floor
                {
                    b1 = final.pop();
                    b2 = (float)Math.Floor(b1);
                    final.push(b2);
                }
                else if (temp.a[i] == -10021)// ceiling
                {
                    b1 = final.pop();
                    b2 = (float)Math.Ceiling(b1);
                    final.push(b2);
                }
                else if (temp.a[i] == -10022)// fraction
                {
                    b1 = final.pop();
                    b2 = (float)(b1 - Math.Floor(b1));
                    final.push(b2);
                }
                else if (temp.a[i] == -10016)// asin
                {
                    b1 = final.pop();
                    b2 = (float)Math.Asin(b1);
                    final.push(b2);
                }
                else if (temp.a[i] == -10017)// acos
                {
                    b1 = final.pop();
                    b2 = (float)Math.Acos(b1);
                    final.push(b2);
                }
                else if (temp.a[i] == -10018)// atan
                {
                    b1 = final.pop();
                    b2 = (float)Math.Atan(b1);
                    final.push(b2);
                }
                else if (temp.a[i] == -10009)// cos
                {
                    b1 = final.pop();
                    b2 = (float)Math.Cos(b1);
                    final.push(b2);
                }
                else if (temp.a[i] == -10010)// tan
                {
                    b1 = final.pop();
                    b2 = (float)Math.Tan(b1);
                    final.push(b2);
                }
                else if (temp.a[i] == -10011)// log
                {
                    b1 = final.pop();
                    if (b1 <= 0)
                        b2 = -10000;
                    else
                        b2 = (float)Math.Log10(b1);
                    final.push(b2);
                }
                else if (temp.a[i] == -10012)// ln
                {
                    b1 = final.pop();
                    if (b1 <= 0)
                        b2 = -10000;
                    else
                        b2 = (float)Math.Log10(b1);
                    final.push(b2);
                }
                else if (temp.a[i] == -10013)// antilog
                {
                    b1 = final.pop();
                    b2 = (float)Math.Pow(10, b1);
                    final.push(b2);
                }
                else if (temp.a[i] == -10014)// antiln
                {
                    b1 = final.pop();
                    b2 = (float)Math.Pow(2.17, b1);
                    final.push(b2);
                }
                else if (temp.a[i] == -10015)// root
                {
                    b1 = final.pop();
                    b2 = (float)Math.Sqrt(b1);
                    final.push(b2);
                }
                else
                    final.push(temp.a[i]);
            }
            return (float)Math.Round(final.a[0], 3);
        }
        private void Graph_MouseDown(object sender, MouseEventArgs e)
        {
            isMouseDown = true;
            isMouseMove = false;
            xDown = e.X;
            yDown = e.Y;
        }

        private void Graph_MouseUp(object sender, MouseEventArgs e)
        {
            Graphics g = Graph.CreateGraphics();
            g.Clear(GridBKColor);
            isMouseDown = false;
            if (fType == 0)
            {
                if (drawI.Checked)
                    DrawArea(FXpost, g);
                drawGrid(g);
                if (fxEntry.Text!="")
                    DrawGraph(FXpost,g);
                if (drawGX.Checked)
                    DrawGraph(GXpost, GXbrush,g);
                if (drawHX.Checked)
                    DrawGraph(HXpost, HXbrush,g);
            }
            else if(fType==1)
            {
                drawGrid(g);
                if (fxEntry.Text != "" && gxEntry.Text != "")
                    DrawGraphPara(FXpost, GXpost, g);

            }
            else if(fType == 2)
            {
                drawGridPolar(g);
                DrawGraphPolar(FXpost,FXbrush, g);
                if (drawGX.Checked)
                    DrawGraphPolar(GXpost, GXbrush, g);
                if (drawHX.Checked)
                    DrawGraphPolar(HXpost, HXbrush, g);
            }
        }

        int gx = 1000, gy = 550;

        bool isMouseDown = false, isMouseMove = true;



        int zoom = 42;



        void drawGrid(Graphics G)
        {

            Font drawFont = new Font("calibri", zoom / 3, FontStyle.Bold);

            // vertical lines & numbers, positive half
            for (int i = gx / 2 + (int)xShift, j = 0; i < gx; i += zoom, j++)
            {
                G.DrawLine(GrayPen, i, 0, i, gy);
                if (yShift < gy / 2 && yShift > -gy / 2)
                    G.DrawString(j.ToString(), drawFont, brush, i, gy / 2 -
                     yShift);
                else if (yShift > gy / 2)
                    G.DrawString(j.ToString(), drawFont, brush, i, 0);
                else if (yShift < -gy / 2)
```

```
                    G.DrawString(j.ToString(), drawFont, brush, i, gy - 30);
            }
            // vertical lines & numbers, negative half
            for (int i = gx / 2 + (int)xShift, j = 0; i > 0; i -= zoom, j--)
            {
                G.DrawLine(GrayPen, i, 0, i, gy);
                if (yShift < gy / 2 && yShift > -gy / 2)
                    G.DrawString(j.ToString(), drawFont, brush, i, gy / 2 -
                     yShift);
                else if (yShift > gy / 2)
                    G.DrawString(j.ToString(), drawFont, brush, i, 0);
                else if (yShift < -gy / 2)
                    G.DrawString(j.ToString(), drawFont, brush, i, gy - 30);

            }
            // horizontal lines & numbers, upper half
            for (int i = gy / 2 - (int)yShift, j = 0; i > 0; i -= zoom, j++)
            {
                G.DrawLine(GrayPen, 0, i, gx, i);
                if (xShift < gy / 2 && xShift > -gy / 2)
                    G.DrawString(j.ToString(), drawFont, brush, gx / 2 + xShift,
                     i);
                else if (xShift < -gx / 2)
                    G.DrawString(j.ToString(), drawFont, brush, 0, i);
                else if (xShift > gx / 2)
                    G.DrawString(j.ToString(), drawFont, brush, gx - 30, i);
            }
            // horizontal lines & numbers, lower half
            for (int i = gy / 2 - (int)yShift, j = 0; i < gy; i += zoom, j--)
            {
                G.DrawLine(GrayPen, 0, i, gx, i);
                if (xShift < gy / 2 && xShift > -gy / 2)
                    G.DrawString(j.ToString(), drawFont, brush, gx / 2 + xShift,
                     i);
                else if (xShift < -gx / 2)
                    G.DrawString(j.ToString(), drawFont, brush, 0, i);
                else if (xShift > gx / 2)
                    G.DrawString(j.ToString(), drawFont, brush, gx - 30, i);
            }
            //drawing axes

            if (yShift < gy / 2 && yShift > -gy / 2)
                G.DrawLine(axisPen, 0, gy / 2 - (int)yShift, gx, gy / 2 - (int)
                  yShift);//x-axis
            else if (yShift > gy / 2)
                G.DrawLine(axisPen, 0, 30, gx, 30);
            else if (yShift < -gy / 2)
                G.DrawLine(axisPen, 0, gy - 30, gx, gy - 30);

            if (xShift < gx / 2 && xShift > -gx / 2)
                G.DrawLine(axisPen, gx / 2 + (int)xShift, 0, gx / 2 + (int)xShift,
                  gy);//y-axis
            else if (xShift > gx / 2)
                G.DrawLine(axisPen, gx - 30, 0, gx - 30, gy);
            else if (xShift < -gx / 2)
                G.DrawLine(axisPen, 30, 0, 30, gy);
```

```
        }
        void drawGrid3D(Graphics G)
        {
            float angleX = 55 / 14.0F, angleY = 0;//(5*pie / 4)
            G.DrawLine(axisPen, 0, (float)(gy / 2.0 + Math.Tan(angleX) * gx / 2.0),
               gx, (float)(gy / 2.0 - Math.Tan(angleX) * gx / 2.0));//x - axis
            G.DrawLine(axisPen, gx / 2, 0, gx / 2, gy);//z - axis
            G.DrawLine(axisPen, 0, (float)(gy / 2.0 + Math.Tan(angleY) * gx / 2.0),
               gx, (float)(gy / 2.0 - Math.Tan(angleY) * gx / 2.0));//y - axis

        }

        void drawGridPolar(Graphics G)
        {

            G.Clear(GridBKColor);
            Font drawFont = new Font("calibri", zoom / 3, FontStyle.Bold);

            // circles & horizontal numbers, positive half
            for (int i = gx / 2 + (int)xShift, j = 0,k=0; i < gx; i += zoom, j++,k
              +=2*zoom)
            {
                G.DrawEllipse(GrayPen, gx / 2 + xShift-k/2, gy / 2 - yShift - k/2,
                  k, k);
                G.DrawString(j.ToString(), drawFont, brush, i, gy / 2 - yShift);

            }
            //  horizontal numbers, negative half
            for (int i = gx / 2 + (int)xShift, j = 0; i > 0; i -= zoom, j--)

                G.DrawString(j.ToString(), drawFont, brush, i, gy / 2 - yShift);

            // virtical numbers, upper half
            for (int i = gy / 2 - (int)yShift, j = 0; i > 0; i -= zoom, j++)

                G.DrawString(j.ToString(), drawFont, brush, gx / 2 + xShift, i);

            // vertical numbers, lower half
            for (int i = gy / 2 - (int)yShift, j = 0; i < gy; i += zoom, j--)

                G.DrawString(j.ToString(), drawFont, brush, gx / 2 + xShift, i);

            //drawing axes
            G.DrawLine(axisPen, 0, gy / 2 - (int)yShift, gx, gy / 2 - (int)
              yShift);//x-axis
            G.DrawLine(axisPen, gx / 2 + (int)xShift, 0, gx / 2 + (int)xShift,
              gy);//y-axis
            //drawing tilted lines
            float root3 = (float)Math.Sqrt(3);
            G.DrawLine(GrayPen, 0, (gy + gx /root3 ) / 2, gx, (gy - gx / root3) /
              2);//line at tan(30)
            G.DrawLine(GrayPen, 0, (gy - gx / root3) / 2, gx, (gy + gx / root3) /
              2);//line at tan(150)
            G.DrawLine(GrayPen, (gx + gy / root3) / 2, 0, (gx - gy / root3) / 2,
              gy);//line at tan(60)
            G.DrawLine(GrayPen, (gx - gy / root3) / 2, 0, (gx + gy / root3) / 2,  ↩
              gy);//line at tan(120)
        }

        void DrawGraph(stack post, Graphics G)
        {
            float position,value,y, yNew,yy,xOld=-5001,yOld=-5001;
            float domainStart, domainEnd, area=0;

            pointList.Items.Clear();
            //drawing +ve half
            for (position = gx / 2 + xShift, value = 0; position <= gx; position+  ↩
              +,value+=1/(float)zoom )
            {
                y = pToIn(post, value);
                yNew = gy / 2 - yShift - y * zoom;

                double distance = Math.Sqrt((xOld - position) * (xOld - position) +↩
                   (yOld - yNew) * (yOld - yNew));
                if (distance < zoom*3)
                    G.DrawLine(FXPen, xOld, yOld, position, yNew);
                xOld = position;
                yOld = yNew;
                //calculating area
                if (y < 0)
                    y = -y;
                area += y / (float)zoom;
            }
            xOld = yOld = -5001;
            domainEnd = (float)Math.Round(value,3);
            for (position = gx / 2 + xShift, value = 0; position >= 0; position--,  ↩
              value -=1 / (float)zoom)
            {
                y = pToIn(post, value);

                yNew = gy / 2 - yShift - y * zoom;
                double distance = Math.Sqrt((xOld - position) * (xOld - position) +↩
                   (yOld - yNew) * (yOld - yNew));
                if (distance < zoom * 3)
                    G.DrawLine(FXPen, xOld, yOld, position, yNew);
                xOld = position;
                yOld = yNew;
                //calculating area
                if (y < 0)
                    y = -y;
                area += y / (float)zoom;
            }
            area = (float)Math.Round(area, 3);
            AbsAreaLabel.Text = area.ToString();
            domainStart = (float)Math.Round(value, 3);
            xRange.Text = "X=[" + domainStart.ToString() + "," + domainEnd.ToString↩
              () + "]";//writing current domain of function
            //generating point list
            for(int i=(int)domainStart;i<(int)domainEnd;i++)
            {
                float j = pToIn(post, i);
                if(j!=-10000)
                pointList.Items.Add(i + "                    " + j);
            }
            if(drawYY.Checked)
            {
                for (position = gx / 2 + xShift, value = 0.01F; position <= gx;    ↩
                  position++, value += 1 / (float)zoom)
                {
                    y = pToIn(post, value);
                    yy = (pToIn(post, value + 0.00001F) - y) / 0.00001F;
                    //yy = (float)Math.Round(yy, 3);
                    yNew = gy / 2 - yShift - yy * zoom;
                    G.FillEllipse(YYPen, position, yNew, 3, 3);
                }
                for (position = gx / 2 + xShift, value = 0.01F; position >= 0;     ↩
                  position--, value -= 1 / (float)zoom)
                {
                    y = pToIn(post, value);
                    yy = (pToIn(post, value + 0.00001F) - y) / 0.00001F;
                    //yy = (float)Math.Round(yy, 3);
                    yNew = gy / 2 - yShift - yy * zoom;
                    G.FillEllipse(YYPen, position, yNew, 3, 3);
                }
            }
        }
        void DrawGraphPara(stack postx, stack posty, Graphics G)
        {
            float position, value, y,x,xNew, yNew, xOld = -5001, yOld = -5001;

            pointList.Items.Clear();
            //drawing +ve half
            for (position = gx / 2 + xShift, value = 0; position <= gx; position++,↩
               value += 1 / (float)zoom)
            {
                x = pToIn(postx, value);
                y = pToIn(posty, value);
                xNew = gx / 2 + xShift + x * zoom;
                yNew = gy / 2 - yShift - y * zoom;
                /*if (y < 0.01 && y > -0.01)
                {
                    rootDisplay.Items.Add("0                          " + value);  ↩

                }*/
                //G.DrawEllipse(FXPen, xNew, yNew, 1, 1);
                double distance = Math.Sqrt((xOld - xNew) * (xOld - xNew) + (yOld -↩
                   yNew) * (yOld - yNew));
                if (distance < zoom * 3)
                    G.DrawLine(FXPen, xOld, yOld, xNew, yNew);
                xOld = xNew;
                yOld = yNew;
            }
            xOld = yOld = -5001;
            for (position = gx / 2 + xShift, value = 0; position >= 0; position--, ↩
              value -= 1 / (float)zoom)
            {
                x = pToIn(postx, value);
                y = pToIn(posty, value);
                /*if (y < 0.01 && y > -0.01 && value!=0)
                {
                    rootDisplay.Items.Add("0                          " + value);  ↩

                }*/
                xNew = gx / 2 + xShift + x * zoom;
                yNew = gy / 2 - yShift - y * zoom;
                double distance = Math.Sqrt((xOld - xNew) * (xOld - xNew) + (yOld -↩
                   yNew) * (yOld - yNew));
                if (distance < zoom * 3)
                    G.DrawLine(FXPen, xOld, yOld, xNew, yNew);
                xOld = xNew;
                yOld = yNew;
            }
        }
            void DrawGraph(stack post,Brush b, Graphics G)
            {
            float position, value, y, yNew, xOld = -5001, yOld = -5001;
            Pen temp = new Pen(b,3);
            //drawing +ve half
            for (position = gx / 2 + xShift, value = 0; position <= gx; position++,↩
               value += 1 / (float)zoom)
            {
                y = pToIn(post, value);
                yNew = gy / 2 - yShift - y * zoom;
                //G.FillEllipse(b, position, yNew, 4, 4);
                double distance = Math.Sqrt((xOld - position) * (xOld - position) +↩
                   (yOld - yNew) * (yOld - yNew));
                if (distance < zoom * 3)
                    G.DrawLine(temp, xOld, yOld, position, yNew);
                xOld = position;
                yOld = yNew;
            }
            xOld = yOld = -5001;
            for (position = gx / 2 + xShift, value = 0; position >= 0; position--, ↩
              value -= 1 / (float)zoom)
            {
                y = pToIn(post, value);
                yNew = gy / 2 - yShift - y * zoom;
                //G.FillEllipse(b, position, yNew, 4, 4);
                double distance = Math.Sqrt((xOld - position) * (xOld - position) +↩
                   (yOld - yNew) * (yOld - yNew));
                if (distance < zoom * 3)
                    G.DrawLine(temp, xOld, yOld, position, yNew);
                xOld = position;
                yOld = yNew;
            }
            }

        void DrawGraphInverted(stack post, Brush b, Graphics G)
        {
            float position, value, y, yNew, xOld = -5001, yOld = -5001;
            Pen temp = new Pen(b, 3);
            //drawing +ve half
            for (position = gx / 2 + xShift, value = 0; position <= gx; position++,↩
                value += 1 / (float)zoom)
            {
                y = pToIn(post, value);
                yNew = gy / 2 - yShift - y * zoom;
                //G.FillEllipse(b, position, yNew, 4, 4);
                double distance = Math.Sqrt((xOld - position) * (xOld - position) +↩
                   (yOld - yNew) * (yOld - yNew));
                if (distance < zoom * 3)
                    G.DrawLine(temp, xOld, yOld, position, yNew);
                xOld = position;
                yOld = yNew;
            }
            xOld = yOld = -5001;
            for (position = gx / 2 + xShift, value = 0; position >= 0; position--, ↩
              value -= 1 / (float)zoom)
            {
                y = pToIn(post, value);
                yNew = gy / 2 - yShift - y * zoom;
                //G.FillEllipse(b, position, yNew, 4, 4);
                double distance = Math.Sqrt((xOld - position) * (xOld - position) +↩
                   (yOld - yNew) * (yOld - yNew));
                if (distance < zoom * 3)
                    G.DrawLine(temp, xOld, yOld, position, yNew);
                xOld = position;
                yOld = yNew;
            }
        }
        void DrawGraph3D(stack post, Brush b, Graphics G)
        {
            float position,posY, value,valueY,x, y,z, yNew, xOld = -5001, yOld =   ↩
              -5001;
            Pen temp = new Pen(b, 3);
            //drawing +ve half
            for (position = gx / 2, value = 0; position <= gx; position++, value +=↩
               1 / (float)zoom)
            {
                for (posY = gy / 2 , valueY = 0; posY > 0; posY--, valueY +=1/      ↩
                  (float)zoom)
                {
                    z = pToIn(post, value, valueY);
                    y = posY + (z*zoom) / (float)Math.Sqrt(2);
                    x = position + (z * zoom) / (float)Math.Sqrt(2);
                    G.FillEllipse(b, x, y, 4, 4);
                }
            }
            for (position = gx / 2 + xShift, value = 0; position >= 0; position--, ↩
              value -= 1 / (float)zoom)
            {
                for (posY = gy / 2, valueY = 0; posY < gy; posY++, valueY -= 1 /   ↩
                  (float)zoom)
                {
                    z = pToIn(post, value, valueY);
                    y = posY + (z * zoom) / (float)Math.Sqrt(2);
                    x = position + (z * zoom) / (float)Math.Sqrt(2);
                    G.FillEllipse(b, x, y, 4, 4);
                }
            }
        }
        void DrawArea(stack post, Graphics G)
        {
            float position, value, y, yNew;
            //drawing +ve half
            for (position = gx / 2 + xShift, value = 0; position <= gx; position++,⤶
               value += 1 / (float)zoom)
            {
                y = pToIn(post, value);
                yNew = gy / 2 - yShift - y * zoom;
                if (yNew > 0 && yNew < gy)
                    G.DrawLine(AreaPen, position, gy / 2 - yShift, position, yNew);
            }
            for (position = gx / 2 + xShift, value = 0; position >= 0; position--, ⤶
              value -= 1 / (float)zoom)
            {
                y = pToIn(post, value);
                yNew = gy / 2 - yShift - y * zoom;
                if (yNew > 0 && yNew < gy)
                    G.DrawLine(AreaPen, position, gy / 2 - yShift, position, yNew);
            }
        }

        void DrawGraphPolar(stack post,Brush b, Graphics G)
        {
            float position, value, y, yNew,r,x,xNew;
            //drawing +ve half
            for (position = gx / 2 + xShift, value = 0; position <= gx; position++,⤶
               value += 1 / (float)zoom)
            {
                r = pToIn(post, value);
                x = r * (float)Math.Cos(value);
                y = r * (float)Math.Sin(value);
                yNew = gy / 2 - yShift - y * zoom;
                xNew = gx / 2 + xShift + x * zoom;
                G.FillEllipse(b, xNew, yNew, 4, 4);
            }
        }

       bool isValidExp(string s)
        {
            bool r;
            int l, c=0;
            l = s.Length;
            for (int i = 0; i < l; i++)
            {
                if (s.Substring(i, 1) == "(")
                    c++;
                else if (s.Substring(i, 1) == ")")
                {
                    c--;
                    if (c < 0)
                        break;
                }
            }
            if (c == 0)
                r = true;
            else
                r = false;
            return r;
        }

        private void antilnButton_Click(object sender, EventArgs e)
        {
            insert("antiln");
        }


        int FlabelCount = 1;
        int MaxF;

        void writeFn(string fn)
        {
            for (int i = 14; i > 0; i--)
            fnHistory[i] = fnHistory[i - 1];
            fnHistory[0] = fn;
            File.WriteAllLines(path, fnHistory);
        }
        void insert(string input)
        {

            if (FlabelCount == 1)
            {
                fxEntry.Text += input;
            }
            else if (FlabelCount == 2)
            {
                gxEntry.Text += input;
            }
            else if (FlabelCount == 3)
            {
                hxEntry.Text += input;
            }
        }


        private void clearButton_Click_1(object sender, EventArgs e)
        {
            fxEntry.Text = "";
            FXinput.top = -1;
            FXpost.top = -1;
            //isFXinputEmpty = true;
        }

        private void one_Click_1(object sender, EventArgs e)
        {
            insert("1");
        }

        private void nine_Click_1(object sender, EventArgs e)
        {
            insert("9");
        }

        private void aButton_Click(object sender, EventArgs e)
        {
            insert("a");
        }

        private void pieButton_Click(object sender, EventArgs e)
        {
            insert("π");
        }

        private void eButton_Click(object sender, EventArgs e)
        {
            insert("e");
        }

        private void pointButton_Click(object sender, EventArgs e)
        {
            insert(".");
        }

        private void antilogButton_Click(object sender, EventArgs e)
        {
            insert("antilog");
        }

        private void antilnButton_Click_1(object sender, EventArgs e)
        {
            insert("antiln");
        }

        private void logButton_Click(object sender, EventArgs e)
        {
            insert("log");
        }

        private void xVariable_Click_1(object sender, EventArgs e)
        {
            insert(xVariable.Text);
        }

        private void lnButton_Click(object sender, EventArgs e)
        {
            insert("ln");
        }

        private void zero_Click(object sender, EventArgs e)
        {
            insert("0");
        }

        private void power_Click(object sender, EventArgs e)
        {
            insert("^");
        }

        private void tanButton_Click(object sender, EventArgs e)
        {
            insert("tan");
        }

        private void bracketClose_Click(object sender, EventArgs e)
        {
            insert(")");
        }

        private void divide_Click(object sender, EventArgs e)
        {
            insert("/");
        }

        private void multiply_Click(object sender, EventArgs e)
        {
            insert("*");
        }

        private void minus_Click(object sender, EventArgs e)
        {
            insert("-");
        }

        private void plus_Click(object sender, EventArgs e)
        {
            insert("+");
        }

        private void cosButton_Click(object sender, EventArgs e)
        {
            insert("cos");
        }
        private void root_Click(object sender, EventArgs e)
        {
            insert("root");
        }
        private void absButton_Click(object sender, EventArgs e)
        {
            insert("abs");
        }
        private void asinButton_Click(object sender, EventArgs e)
        {
            insert("asin");
        }
        private void acosButton_Click(object sender, EventArgs e)
        {
            insert("acos");
        }
        private void atanButton_Click(object sender, EventArgs e)
        {
            insert("atan");
        }
        private void floor_Click(object sender, EventArgs e)
        {
            insert("floor");
        }
        private void fraction_Click(object sender, EventArgs e)
        {
            insert("frac");
        }
        private void ceiling_Click(object sender, EventArgs e)
        {
            insert("ceil");
        }
        private void three_Click(object sender, EventArgs e)
        {
            insert("3");
        }

        private void two_Click(object sender, EventArgs e)
        {
            insert("2");
        }

        private void sinButton_Click(object sender, EventArgs e)
        {
            insert("sin");
        }

        private void six_Click(object sender, EventArgs e)
        {
            insert("6");
        }

        private void five_Click(object sender, EventArgs e)
        {
            insert("5");
        }

        private void four_Click(object sender, EventArgs e)
        {
            insert("4");
        }

        private void bracketOpen_Click(object sender, EventArgs e)
        {
            insert("(");
        }

        private void eight_Click(object sender, EventArgs e)
        {
            insert("8");
        }

        private void seven_Click(object sender, EventArgs e)
        {
            insert("7");
        }
        private void drawButton_Click_1(object sender, EventArgs e)
        {
            if (isValidExp(fxEntry.Text))
            {
                Graphics g = Graph.CreateGraphics();
                g.Clear(GridBKColor);
                stringToStack(FXinput, fxEntry.Text);
                inToP(FXinput, FXpost);
                writeFn(fxEntry.Text);
                outputBox.Visible = true;
                inputBox.Visible = false;
                if (fType == 0)
                {
                    if (drawI.Checked)
                        DrawArea(FXpost, g);
                    drawGrid(g);
                    DrawGraph(FXpost, g);
                    if (drawGX.Checked && isValidExp(gxEntry.Text))
                    {
                        stringToStack(GXinput, gxEntry.Text);
                        inToP(GXinput, GXpost);
                        DrawGraph(GXpost, GXbrush, g);
                    }
                    if (drawHX.Checked && isValidExp(hxEntry.Text))
                    {
                        stringToStack(HXinput, hxEntry.Text);
                        inToP(HXinput, HXpost);
                        DrawGraph(HXpost, HXbrush, g);
                    }
                }
                else if (fType == 3)
                {
                    //if (drawI.Checked)
                    //    DrawArea(FXpost, g);
                    drawGrid(g);
                    DrawGraphInverted(FXpost, FXbrush, g);
                    if (drawGX.Checked && isValidExp(gxEntry.Text))
                    {
                        stringToStack(GXinput, gxEntry.Text);
                        inToP(GXinput, GXpost);
                        DrawGraphInverted(GXpost, GXbrush, g);
                    }
                    if (drawHX.Checked && isValidExp(hxEntry.Text))
                    {
                        stringToStack(HXinput, hxEntry.Text);
                        inToP(HXinput, HXpost);
                        DrawGraphInverted(HXpost, HXbrush, g);
                    }
                }
                else if (fType == 1)
                {
                    stringToStack(GXinput, gxEntry.Text);
                    inToP(GXinput, GXpost);
                    drawGrid(g);
                    DrawGraphPara(FXpost, GXpost, g);
                }
                else if (fType == 2)
                {
                    xShift = yShift = 0;
                    drawGridPolar(g);
                    DrawGraphPolar(FXpost, FXbrush, g);
                    if (drawGX.Checked && isValidExp(gxEntry.Text))
                    {
                        //GXinput.push(-10003);// )
                        stringToStack(GXinput, gxEntry.Text);
                        inToP(GXinput, GXpost);
                        DrawGraphPolar(GXpost, GXbrush, g);
                    }
                    if (drawHX.Checked && isValidExp(hxEntry.Text))
                    {
                        //HXinput.push(-10003);// )
                        stringToStack(HXinput, hxEntry.Text);
                        inToP(HXinput, HXpost);
                        DrawGraphPolar(HXpost, HXbrush, g);
                    }
                }
                else if (fType == 3)
                {
                    drawGrid3D(g);
                }
                xShift = yShift = 0;
                zoomScroll.Focus();
                zoomScroll.Value = 4;
            }
        }

        private void zoomScroll_Scroll_1(object sender, EventArgs e)
        {
            Graphics g = Graph.CreateGraphics();
            g.Clear(GridBKColor);
            int z = zoomScroll.Value;
            if (z != zoom * 10)
            {
                zoom = 10 + z * 8;
                if (fType == 0)
                {
                    if (drawI.Checked)
                        DrawArea(FXpost, g);
                    drawGrid(g);
                    if (fxEntry.Text!="")
                        DrawGraph(FXpost,g);
                    if (drawGX.Checked)
                        DrawGraph(GXpost, GXbrush,g);
                    if (drawHX.Checked)
                        DrawGraph(HXpost, HXbrush,g);
                }
                else if (fType == 3)
                {
                    //if (drawI.Checked)
                    //    DrawArea(FXpost, g);
                    drawGrid(g);
                    if (fxEntry.Text != "")
                        DrawGraphInverted(FXpost, FXbrush, g);
                    if (drawGX.Checked)
                        DrawGraphInverted(GXpost, GXbrush, g);
                    if (drawHX.Checked)
                        DrawGraphInverted(HXpost, HXbrush, g);
                }
                else if(fType ==1)
                {
                    drawGrid(g);
                    if (fxEntry.Text != ""&& gxEntry.Text!="")
                        DrawGraphPara(FXpost,GXpost, g);
                }
                else if(fType == 2)
                {
                    drawGridPolar(g);
                    if (fxEntry.Text!="")
                        DrawGraphPolar(FXpost,FXbrush, g);
                    if (drawGX.Checked)
                        DrawGraphPolar(GXpost, GXbrush, g);
                    if (drawHX.Checked)
                        DrawGraphPolar(HXpost, HXbrush, g);
                }
            }
        }

        private void centerButton_Click_1(object sender, EventArgs e)
        {
            Graphics g = Graph.CreateGraphics();
            g.Clear(GridBKColor);
            xShift = yShift = 0;
                drawGrid(g);
                DrawGraph(FXpost,g);
                if (drawGX.Checked)
                    DrawGraph(GXpost, GXbrush,g);
                if (drawHX.Checked)
                    DrawGraph(HXpost, HXbrush,g);
            if (drawI.Checked)
                DrawArea(FXpost, g);

        }

        private void locateButton_Click_1(object sender, EventArgs e)
        {
            Graphics g = Graph.CreateGraphics();
            if (textBoxXEntry.Text != "" && textBoxXEntry.Text != "-")
            {
                if (fType == 0)
                {
                    xShift = -float.Parse(textBoxXEntry.Text) * zoom;
                    yShift = -float.Parse(labelYOutput.Text) * zoom;
                    g.Clear(GridBKColor);
                    if (drawI.Checked)
                        DrawArea(FXpost, g);
                    drawGrid(g);
                    DrawGraph(FXpost,g);
                    if (drawGX.Checked)
                        DrawGraph(GXpost,GXbrush, g);
                    if (drawHX.Checked)
                        DrawGraph(HXpost,HXbrush, g);
                }
                else if (fType == 2)
                {
                    stack post = FXpost;
                    float r = pToIn(post, float.Parse(textBoxXEntry.Text) * zoom);
                    float x = r * (float)Math.Cos(float.Parse(textBoxXEntry.Text) *
                      zoom);
                    float y = r * (float)Math.Sin(float.Parse(textBoxXEntry.Text) *
                      zoom);
                    xShift = -x;
                    yShift = -y;
                    drawGridPolar(g);
                    if (fxEntry.Text != "")
                        DrawGraphPolar(FXpost,FXbrush, g);
                    if (drawGX.Checked)
                        DrawGraphPolar(GXpost,GXbrush, g);
                    if (drawHX.Checked)
                        DrawGraphPolar(HXpost,HXbrush, g);
                }
            }
        }

        private void textBoxXEntry_TextChanged_1(object sender, EventArgs e)
        {
            if (textBoxXEntry.Text != "" && textBoxXEntry.Text != "-")
            {
                float x = float.Parse(textBoxXEntry.Text);
                float n = pToIn(FXpost, x);
                labelYOutput.Text = n.ToString();
                float y2 = (pToIn(FXpost, x + 0.000001F) - n) / 0.000001F;
                labelYYOutput.Text = y2.ToString();
            }
            else
            {
                labelYYOutput.Text = "";
                labelYOutput.Text = "";
            }
        }

        private void resetButton_Click_1(object sender, EventArgs e)
        {
            Graphics g = Graph.CreateGraphics();
            FXinput.top = GXinput.top = HXinput.top = -1;
            FXpost.top = GXpost.top = HXpost.top = -1;
            inputBox.Visible = true;
            outputBox.Visible = false;
            if (fType == 0 || fType == 1)
                drawGrid(g);
            else if (fType == 2)
                drawGridPolar(g);
            textBoxXEntry.Text = labelYOutput.Text = labelYYOutput.Text =
              AbsAreaLabel.Text = "";
```

```
            fxEntry.Enabled = gxEntry.Enabled = hxEntry.Enabled = true;
            displayHistory.Items.Clear();
            for (int i = 0; i < 15; i++)
                displayHistory.Items.Add(fnHistory[i]);
        }

        private void aEntry_TextChanged_1(object sender, EventArgs e)
        {
            if (aEntry.Text != "")
                a = float.Parse(aEntry.Text);
        }

        private void ThemeBox_SelectedIndexChanged_1(object sender, EventArgs e)
        {
            Graphics g = Graph.CreateGraphics();
            if (ThemeBox.SelectedItem.ToString() == "White")
            {
                Form1.ActiveForm.BackColor = Color.White;
                zoomScroll.BackColor = label8.BackColor = textBoxXEntry.BackColor =↩
                   Color.White;
                axisPen.Color = label1.ForeColor = brush.Color = Color.Black;
                FxLabel.ForeColor = fxEntry.ForeColor = label2.ForeColor =          ↩
                  label3.ForeColor = labelTheme.ForeColor = labelFType.ForeColor = ↩
                  labelYOutput.ForeColor = Color.DarkGray;
                gxLabel.ForeColor = hxLabel.ForeColor = gxEntry.ForeColor =         ↩
                  hxEntry.ForeColor = label5.ForeColor = labelYYOutput.ForeColor = ↩
                  Color.FromArgb(30,30,30);
                GridBKColor = Color.FromArgb(250,250,250);
                drawYY.ForeColor = drawGX.ForeColor = drawHX.ForeColor =            ↩
                  Color.Black;
                graphBox.BackColor = points.BackColor = input.BackColor =           ↩
                  settings.BackColor = Color.White;
                label6.BackColor = label7.BackColor = label9.BackColor =            ↩
                  label10.BackColor = Color.White;
                FXbrush.Color = Color.FromArgb(00, 00, 64);
            }
            if (ThemeBox.SelectedItem.ToString() == "Black")
            {
                Form1.ActiveForm.BackColor = Color.FromArgb(60, 60, 60);
                zoomScroll.BackColor = label8.BackColor = textBoxXEntry.BackColor =↩
                   Color.Black;
                axisPen.Color = label1.ForeColor = brush.Color = Color.White;
                FxLabel.ForeColor = fxEntry.ForeColor = label2.ForeColor =          ↩
                  label3.ForeColor = labelTheme.ForeColor = labelFType.ForeColor = ↩
                  labelYOutput.ForeColor = Color.FromArgb(30,30,30);
                GridBKColor = Color.DarkGray;
                FXbrush.Color = Color.Black;
            }
            drawGrid(g);
        }

        private void FnTypeBox_SelectedIndexChanged(object sender, EventArgs e)
        {
            fType = FnTypeBox.SelectedIndex;
            if(fType == 1)
            {
                hxEntry.Visible = false;
                hxLabel.Visible = false;
                FxLabel.Text = "X(t) :";
                gxLabel.Text = "Y(t) :";
                xVariable.Text = "t";
            }
            else if (fType == 0)
            {
                hxEntry.Visible = true;
                hxLabel.Visible = true;
                FxLabel.Text = "f(x) :";
                gxLabel.Text = "g(x) :";
                hxLabel.Text = "h(x) :";
                xVariable.Text = "x";
            }
            else if (fType == 2)
            {
                hxEntry.Visible = true;
                hxLabel.Visible = true;
                FxLabel.Text = "r1(t) :";
                gxLabel.Text = "r2(t) :";
                hxLabel.Text = "r3(t) :";
                xVariable.Text = "t";
            }
            else if (fType == 3)
            {
                hxEntry.Visible = true;
                hxLabel.Visible = true;
                FxLabel.Text = "f(y) :";
                gxLabel.Text = "g(y) :";
                hxLabel.Text = "h(y) :";
                xVariable.Text = "y";
            }
        }
        void stringToStack(stack input, string name)
        {
            input.push(-10002);
            int l = name.Length;
            for(int i=0;i<l;)
            {
                if (name.Substring(i, 1) == ".")
                    input.push(-9999);
                else if (name.Substring(i, 1) == "x")
                    input.push(-10001);
                else if (name.Substring(i, 1) == "(")
                    input.push(-10002);
                else if (name.Substring(i, 1) == ")")
                    input.push(-10003);
                else if (name.Substring(i, 1) == "+")
                    input.push(-10004);
                else if (name.Substring(i, 1) == "-")
                    input.push(-10005);
                else if (name.Substring(i, 1) == "/")
                    input.push(-10007);
                else if (name.Substring(i, 1) == "*")
                    input.push(-10006);
                else if (name.Substring(i, 1) == "^")
                    input.push(-9998);
                else if (name.Substring(i, 1) == "e")
                    input.push(2.718F);
                else if (name.Substring(i, 1) == "π")
                    input.push(22 / (7F));
                else if (name.Substring(i, 1) == "a")
                {
                    if (name.Substring(i, 3) == "abs")
                    { input.push(-10020); i += 2; }
                    else if (name.Substring(i, 4) == "asin")
                    { input.push(-10016); i += 3; }
                    else if (name.Substring(i, 4) == "acos")
                    { input.push(-10017); i += 3; }
                    else if (name.Substring(i, 6) == "antiln")
                    { input.push(-10014); i += 5; }
                    else if (name.Substring(i, 7) == "antilog")
                    { input.push(-10013); i += 6; }
                    else input.push(a);
                }
                else if (name.Substring(i, 1) == "c")
                {
                    if (name.Substring(i, 4) == "ceil")
                    { input.push(-10021); i += 3; }
                    else if (name.Substring(i, 3) == "cos")
                    { input.push(-10009); i += 2; }
                }
                else if (name.Substring(i, 1) == "f")
                {
                    if (name.Substring(i, 5) == "floor")
                    { input.push(-10019); i += 4; }
                    else if (name.Substring(i, 4) == "frac")
                    { input.push(-10022); i += 3; }
                }
                else if (name.Substring(i, 1) == "l")
                {
                    if (name.Substring(i, 2) == "ln")
                    { input.push(-10012); i++; }
                    else if (name.Substring(i, 3) == "log")
                    { input.push(-10011); i += 2; }
                }
                else if (name.Substring(i, 1) == "r")
                {
                    if (name.Substring(i, 4) == "root")
                    { input.push(-10015); i += 3; }
                }
                else if (name.Substring(i, 1) == "s")
                {
                    if (name.Substring(i, 3) == "sin")
                    { input.push(-10008); i += 2; }
                }
                else if (name.Substring(i, 1) == "t")
                {   if (l - i > 2)
                    {
                        if (name.Substring(i, 3) == "tan")
                        { input.push(-10010); i += 2; }
                    }
                    else
                        input.push(-10001);
                }
                else
                {
                    int z = int.Parse(name.Substring(i, 1));
                    if(z>=0&&z<=9)
                    input.push(z);
                }
                ++i;
            }
            input.push(-10003);
        }

        private void fxEntry_MouseClick(object sender, MouseEventArgs e)
        {
            FlabelCount = 1;
        }

        private void gxEntry_MouseClick(object sender, MouseEventArgs e)
        {
            FlabelCount = 2;
        }

        private void hxEntry_MouseClick(object sender, MouseEventArgs e)
        {
            FlabelCount = 3;
        }
        void Examples(string fn, string type)
        {
            Graphics g = Graph.CreateGraphics();
            g.Clear(GridBKColor);
            fxEntry.Text = fn;
            stringToStack(FXinput, fn);
            inToP(FXinput, FXpost);
            outputBox.Visible = true;
            inputBox.Visible = false;
            if(type=="normal")
            {
                fType = 0;
                drawGrid(g);
                DrawGraph(FXpost, g);
            }
            else if(type == "polar")
            {
                fType = 2;
                drawGridPolar(g);
                DrawGraphPolar(FXpost,FXbrush, g);
            }
            fxEntry.Enabled = gxEntry.Enabled = hxEntry.Enabled = false;
        }
        private void nf1_Click(object sender, EventArgs e)
        {
            Examples(nf1.Text, "normal");
        }
        private void nf2_Click(object sender, EventArgs e)
        {
            Examples(nf2.Text, "normal");
        }

        private void nf3_Click(object sender, EventArgs e)
        {
            Examples(nf3.Text, "normal");
        }

        private void nf4_Click(object sender, EventArgs e)
        {
            Examples(nf4.Text, "normal");
        }

        private void pf1_Click(object sender, EventArgs e)
        {
            Examples(pf1.Text, "polar");
        }

        private void pf2_Click(object sender, EventArgs e)
        {
            Examples(pf2.Text, "polar");
        }

        private void pf3_Click(object sender, EventArgs e)
        {
            Examples(pf3.Text, "polar");
        }

        private void displayHistory_SelectedIndexChanged(object sender, EventArgs  ↩
          e)
        {
            Examples(displayHistory.Text, "normal");
        }

        void CalcArea(float a,float b)
        {
            float value, y,area=0;
            for (value = a; value <= b; value += 1 / (float)zoom)
            {
                y = pToIn(FXpost, value);
                if (y < 0)
                    y = -y;
                area += y / (float)zoom;
            }
            area = (float)Math.Round(area, 3);
            CustomAreaLabel.Text = area.ToString();
        }
        private void customArea_A_TextChanged(object sender, EventArgs e)
        {
            if (customArea_A.Text != "" && customArea_B.Text != "" &&            ↩
              customArea_A.Text != "-" && customArea_B.Text != "-")
            CalcArea(float.Parse(customArea_A.Text), float.Parse                 ↩
              (customArea_B.Text));
        }

        private void customArea_B_TextChanged(object sender, EventArgs e)
        {
            if (customArea_A.Text != "" && customArea_B.Text != "" &&
              customArea_A.Text != "-" && customArea_B.Text != "-")
            CalcArea(float.Parse(customArea_A.Text), float.Parse
              (customArea_B.Text));
        }

        private void fxEntry_TextChanged(object sender, EventArgs e)
        {
            if (isValidExp(fxEntry.Text))
                fxEntry.ForeColor = Color.Navy;
            else
                fxEntry.ForeColor = Color.Red;
        }

        private void gxEntry_TextChanged(object sender, EventArgs e)
        {
            if (isValidExp(gxEntry.Text))
                gxEntry.ForeColor = Color.Navy;
            else
                gxEntry.ForeColor = Color.Red;
        }

        private void hxEntry_TextChanged(object sender, EventArgs e)
        {
            if (isValidExp(hxEntry.Text))
                hxEntry.ForeColor = Color.Navy;
            else
                hxEntry.ForeColor = Color.Red;
        }

        private void saveButton_Click(object sender, EventArgs e)
        {
            Bitmap b = new Bitmap(gx,gy);
            Font f = new Font("calibri", 22, FontStyle.Bold);
            using (Graphics g = Graphics.FromImage(b))
            {
                g.Clear(GridBKColor);
                g.DrawString("f(x) = " + fxEntry.Text, f, FXbrush, 20, 20);
                if(gxEntry.Text!="")
                    g.DrawString("g(x) = " + gxEntry.Text, f, GXbrush, 20, 50);
                if (hxEntry.Text != "")
                    g.DrawString("h(x) = " + hxEntry.Text, f, HXbrush, 20, 80);

                if (fType == 0)
                {
                    if (drawI.Checked)
                        DrawArea(FXpost, g);
                    drawGrid(g);
                    if (fxEntry.Text!="")
                        DrawGraph(FXpost, g);
                    if (drawGX.Checked)
                        DrawGraph(GXpost, GXbrush, g);
                    if (drawHX.Checked)
                        DrawGraph(HXpost, HXbrush, g);
                }
                else if (fType == 2)
                {
                    drawGridPolar(g);
                    if (fxEntry.Text != "")
                        DrawGraphPolar(FXpost,FXbrush, g);
                    if (drawGX.Checked)
                        DrawGraphPolar(GXpost, GXbrush, g);
                    if (drawHX.Checked)
                        DrawGraphPolar(HXpost, HXbrush, g);
                }

                string savedFile = "";
                SFdialog.Filter = "Bitmap Image|*.bmp";
                if (SFdialog.ShowDialog() != DialogResult.Cancel)
                {
                    savedFile = SFdialog.FileName;
                    b.Save(savedFile);
                }
            }
        }

        private void Graph_MouseMove(object sender, MouseEventArgs e)
        {
            if (fType == 0||fType == 1)
            {
                Graphics g = Graph.CreateGraphics();
                xMove = e.X;
                yMove = e.Y;
                float xDiff, yDiff;
                xDiff = xMove - xDown;
                yDiff = 0 - (yMove - yDown);
                if (isMouseDown)
                {
                    g.Clear(GridBKColor);
                    xShift += xDiff / zoom;
                    yShift += yDiff / zoom;
                    drawGrid(g);
                    //isMouseDown = false;
                    //isMouseMove = true;
                }
            }
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            if (File.Exists(path))
            {
                fnHistory = File.ReadAllLines(path);
                for (int i = 0; i < 15; i++)
                    displayHistory.Items.Add(fnHistory[i]);
            }
            else
                File.WriteAllLines(path, fnHistory);
        }
        //bool isFXinputEmpty = true;
        bool isGXinputEmpty = true;
        bool isHXinputEmpty = true;
        private Color GridBKColor = Color.FromArgb(00, 00, 64);

        private void drawButton_Click(object sender, EventArgs e)
        {

        }
        private void xVariable_Click(object sender, EventArgs e)
        {
            insert("x");
        }
    }
}
```