```
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 EventCW1
{
    public partial class frmMain : Form
    {
        public frmMain()
        {
            InitializeComponent();
        }

        private const int numberofstartingitems = 30;
        private const int maxnumberoflistitems = 30;
        private Random random = new Random();

        public bool checkNumberExists(int number)
        {
            for (int position = 0; position < lstNumbers.Items.Count; position++)
            {
                if (Convert.ToInt32(lstNumbers.Items[position]) == number) return true;
            }
            return false;
        }

        public void updateStatus()
        {
            if(lstNumbers.Items.Count > 0)
            {
                lblCountValue.Text = lstNumbers.Items.Count.ToString();
                lblFirstValue.Text = lstNumbers.Items[0].ToString();
                lblLastValue.Text = lstNumbers.Items[lstNumbers.Items.Count - 1].ToString();

                int minimum = Convert.ToInt32(lstNumbers.Items[0]);
                for (int position = 0; position < lstNumbers.Items.Count; position++)
                {
                    if (Convert.ToInt32(lstNumbers.Items[position]) < minimum) minimum = Convert.ToInt32(lstNumbers.Items[position]);

                }
                lblMinimumValue.Text = minimum.ToString();

                int maximum = Convert.ToInt32(lstNumbers.Items[0]);
                for (int position = 0; position < lstNumbers.Items.Count; position++)
                {
                    if (Convert.ToInt32(lstNumbers.Items[position]) > maximum) maximum = Convert.ToInt32(lstNumbers.Items[position]);
                }
                lblMaximumValue.Text = maximum.ToString();

                lblEntriesRemainingValue.Text = (maxnumberoflistitems - lstNumbers.Items.Count).ToString();
                lblMaximumEntriesValue.Text = maxnumberoflistitems.ToString();
            }

            else
            {
                lblCountValue.Text = lstNumbers.Items.Count.ToString();
                lblMinimumValue.Text = "-";
                lblMaximumValue.Text = "-";
                lblFirstValue.Text = "-";
                lblLastValue.Text = "-";
                lblEntriesRemainingValue.Text = maxnumberoflistitems.ToString();
                lblMaximumEntriesValue.Text = maxnumberoflistitems.ToString();
            }

        }

        public void unsortedInsert(int number)
        {
            lstNumbers.Items.Add(number);
        }

        public void sortedInsert(int number)
        {
            int insertposition = 0;

            for (int position = 0; position < lstNumbers.Items.Count; position++)
            {
                if (Convert.ToInt32(lstNumbers.Items[position]) > number)
                {
                    insertposition = position;
                    break;
                }
                else insertposition = lstNumbers.Items.Count;
            }

            lstNumbers.Items.Add(0);

            for (int position = lstNumbers.Items.Count - 1; position >= 0; position--)
            {
                if (position == insertposition)
                {
                    lstNumbers.Items[position] = number;
                    break;
                }
                else lstNumbers.Items[position] = lstNumbers.Items[position -
```

```
                    1];
            }
        }

        public void deleteNumber(int index)
        {
            if (lstNumbers.Items.Count > 0)
            {
                for (int position = index; position < lstNumbers.Items.Count - 1; position++)
                {
                    lstNumbers.Items[position] = lstNumbers.Items[position + 1];
                }
            }

            lstNumbers.Items.RemoveAt(lstNumbers.Items.Count - 1);

            updateStatus();

            if (lstNumbers.Items.Count < maxnumberoflistitems)
            {
                btnInitialise.Enabled = true;
                btnInsert.Enabled = true;
            }

            if (lstNumbers.Items.Count < 2 && optUnsorted.Checked == true) btnShuffle.Enabled = false;

            if (lstNumbers.Items.Count == 0)
            {
                btnSearch.Enabled = false;
                btnClear.Enabled = false;
                grpSearch.Enabled = false;
            }

            lstNumbers.SelectedIndex = -1;

            if (lstNumbers.SelectedIndex == -1)
            {
                btnDelete.Enabled = false;
                picBin.Enabled = false;
            }
        }

        private void frmMain_Load(object sender, EventArgs e)
        {
            picBin.AllowDrop = true;
            updateStatus();
        }

        private void lstNumbers_MouseDown(object sender, MouseEventArgs e)
        {
            if (lstNumbers.Items.Count > 0 && lstNumbers.SelectedIndex != -1)
            {
                btnDelete.Enabled = true;
                picBin.Enabled = true;
                lstNumbers.DoDragDrop(lstNumbers.SelectedIndex.ToString(), DragDropEffects.Copy);
            }
        }

        private void btnInitialise_Click(object sender, EventArgs e)
        {
            int itemcount = lstNumbers.Items.Count;

            for (int position = 0; position < (maxnumberoflistitems - itemcount); position++)
            {
                int number;
                bool numberexists = false;

                do
                {
                    number = random.Next(100 + 1);
                    numberexists = checkNumberExists(number);
                } while (numberexists == true);

                if (optUnsorted.Checked == true) unsortedInsert(number);
                else if (optSorted.Checked == true) sortedInsert(number);

            }

            if (lstNumbers.Items.Count > 0)
            {
                grpSearch.Enabled = true;
                btnSearch.Enabled = true;
                btnClear.Enabled = true;
            }

            if (lstNumbers.Items.Count > 1 && optUnsorted.Checked == true) btnShuffle.Enabled = true;

            if (lstNumbers.Items.Count == maxnumberoflistitems)
            {
                btnInsert.Enabled = false;
                btnInitialise.Enabled = false;
            }

            updateStatus();
        }

        private void btnInsert_Click(object sender, EventArgs e)
        {
            int number;

            try
            {
                number = int.Parse(txtUserInput.Text);
            }
            catch (System.FormatException)
            {
                MessageBox.Show("Input must be a number");
                txtUserInput.Text = "";
                txtUserInput.Focus();
                return;
            }
            catch (System.OverflowException)
            {
                MessageBox.Show("Interger Overflow, Number must be in the range of an unsigned integer (-2,147,483,648 to 2,147,483,647)");
                txtUserInput.Text = "";
                txtUserInput.Focus();
                return;
            }

            if (number < 0 || number > 100)
            {
                MessageBox.Show("Number must be between 0 and 100 inclusive");
                txtUserInput.Text = "";
                txtUserInput.Focus();
                return;
            }

            if (checkNumberExists(number) == true)
            {
                MessageBox.Show("Number must be unique");
                txtUserInput.Text = "";
                txtUserInput.Focus();
                return;
            }

            if (optUnsorted.Checked == true) unsortedInsert(number);
            else if (optSorted.Checked == true) sortedInsert(number);

            if (lstNumbers.Items.Count > 0)
            {
                grpSearch.Enabled = true;
                btnSearch.Enabled = true;
                btnClear.Enabled = true;
            }

            if (lstNumbers.Items.Count > 1 && optUnsorted.Checked == true) btnShuffle.Enabled = true;

            if (lstNumbers.Items.Count == maxnumberoflistitems)
            {
                btnInsert.Enabled = false;
                btnInitialise.Enabled = false;
            }

            txtUserInput.Text = "";
            txtUserInput.Focus();

            updateStatus();
        }

        private void btnShuffle_Click(object sender, EventArgs e)
        {
            int source, destination;

            for (int position = 0; position < lstNumbers.Items.Count; position
              ++)
            {
                do
                {
                    source = random.Next(lstNumbers.Items.Count);
                    destination = random.Next(lstNumbers.Items.Count);
                } while (source == destination);

                object temp = lstNumbers.Items[source];
                lstNumbers.Items[source] = lstNumbers.Items[destination];
                lstNumbers.Items[destination] = temp;
            }

            updateStatus();
        }

        private void btnSearch_Click(object sender, EventArgs e)
        {
            int number;

            try
            {
                number = int.Parse(txtUserInput.Text);
            }
            catch (System.FormatException)
            {
                MessageBox.Show("Input must be a number");
                txtUserInput.Text = "";
                txtUserInput.Focus();
                return;
            }
            catch (System.OverflowException)
            {
                MessageBox.Show("Interger Overflow, Number must be in the
                  range of an unsigned integer (-2,147,483,648 to
                  2,147,483,647)");
                txtUserInput.Text = "";
                txtUserInput.Focus();
                return;
            }

            if (number < 0 || number > 100)
            {
                MessageBox.Show("Number must be between 0 and 100 inclusive");
                txtUserInput.Text = "";
                txtUserInput.Focus();
                return;
            }

            int numberofprobes = 0;
            bool numberfound = false;

            if (optLinear.Checked == true)
            {
                for (int position = 0; position < lstNumbers.Items.Count; position++)
                {
                    numberofprobes++;
                    if (number == Convert.ToInt32(lstNumbers.Items[position]))
                    {
                        numberfound = true;
                        MessageBox.Show("Search query: " + number.ToString() + "\n Found: True \n List index: " + position.ToString() + "\n Number of search probes : " + numberofprobes.ToString());
                        break;
                    }
                    else numberfound = false;
                }

                if (numberfound == false) MessageBox.Show("Search query: " + number.ToString() + "\n Found: False \n Number of search probes : " + numberofprobes.ToString());
            }

            else if(optBinary.Checked == true)
            {
                int searchbegin = 0;
                int searchend = lstNumbers.Items.Count - 1;
                int searchmid;

                while(!(searchbegin > searchend))
                {
                    numberofprobes++;
                    searchmid = (searchbegin + searchend) / 2;

                    if (Convert.ToInt32(lstNumbers.Items[searchmid]) < number) searchbegin = searchmid + 1;
                    else if (Convert.ToInt32(lstNumbers.Items[searchmid]) > number) searchend = searchmid - 1;
                    else if (Convert.ToInt32(lstNumbers.Items[searchmid]) == number)
                    {
                        numberfound = true;
                        MessageBox.Show("Search query: " + number.ToString() + "\n Found: True \n List index: " + searchmid.ToString() + "\n Number of search probes : " + numberofprobes.ToString());
                        break;
                    }
                }

                if (numberfound == false) MessageBox.Show("Search query: " + number.ToString() + "\n Found: False \n Number of search probes : " + numberofprobes.ToString());
            }

```

```csharp
            txtUserInput.Text = "";
            txtUserInput.Focus();
        }

        private void btnClear_Click(object sender, EventArgs e)
        {
            for (int position = lstNumbers.Items.Count - 1; position >=0; position--)
            {
                deleteNumber(position);
            }
        }

        private void btnExit_Click(object sender, EventArgs e)
        {
            DialogResult buttonPressed;

            buttonPressed = MessageBox.Show("Are you sure you want to exit?", "Exit", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

            if (buttonPressed == DialogResult.Yes) this.Close();
        }

        private void btnDelete_Click(object sender, EventArgs e)
        {
            deleteNumber(lstNumbers.SelectedIndex);
        }

        private void picBin_DragEnter(object sender, DragEventArgs e)
        {
            e.Effect = DragDropEffects.Copy;
        }

        private void picBin_DragDrop(object sender, DragEventArgs e)
        {
            int index;

            try
            {
                index = int.Parse(e.Data.GetData(DataFormats.StringFormat).ToString());
            }
            catch
            {
                MessageBox.Show("Input must be a number");
                return;
            }

            if (index < 0 || index >= lstNumbers.Items.Count)
            {
                MessageBox.Show("Input must be a valid list index");
                return;
            }

            deleteNumber(index);
        }

        private void optUnsorted_Click(object sender, EventArgs e)
        {
            if (lstNumbers.Items.Count > 1) btnShuffle.Enabled = true;
            optBinary.Enabled = false;
            optLinear.Select();
        }

        private void optSorted_Click(object sender, EventArgs e)
        {
            object temp;
            bool swap;
            do
            {
                swap = false;

                for (int position = 0; position < lstNumbers.Items.Count - 1; position++)
                {
                    if (Convert.ToInt32(lstNumbers.Items[position]) > Convert.ToInt32(lstNumbers.Items[position + 1]))
                    {
                        temp = lstNumbers.Items[position];
                        lstNumbers.Items[position] = lstNumbers.Items[position + 1];
                        lstNumbers.Items[position + 1] = temp;
                        swap = true;
                    }
                }
            } while (swap);

            optBinary.Enabled = true;
            btnShuffle.Enabled = false;

            updateStatus();
        }
    }
}

```