```
1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel;
4  using System.Data;
5  using System.Drawing;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9  using System.Windows.Forms;
10
11 namespace EventCW1
12 {
13     public partial class frmMain : Form
14     {
15         public frmMain()
16         {
17             InitializeComponent();
18         }
19
20         private const int numberofstartingitems = 30;
21         private const int maxnumberoflistitems = 30;
22         private Random random = new Random();
23
24         public bool checkNumberExists(int number)
25         {
26             for (int position = 0; position < lstNumbers.Items.Count; position
                 ++)
27             {
28                 if (Convert.ToInt32(lstNumbers.Items[position]) == number)
                     return true;
29             }
30             return false;
31         }
32
33         public void updateStatus()
34         {
35             if(lstNumbers.Items.Count > 0)
36             {
37                 lblCountValue.Text = lstNumbers.Items.Count.ToString();
38                 lblFirstValue.Text = lstNumbers.Items[0].ToString();
39                 lblLastValue.Text = lstNumbers.Items[lstNumbers.Items.Count -
                     1].ToString();
40
41                 int minimum = Convert.ToInt32(lstNumbers.Items[0]);
42                 for (int position = 0; position < lstNumbers.Items.Count;
                     position++)
43                 {
44                     if (Convert.ToInt32(lstNumbers.Items[position]) < minimum)
                         minimum = Convert.ToInt32(lstNumbers.Items[position]);
45
46                 }
47                 lblMinimumValue.Text = minimum.ToString();
48
49                 int maximum = Convert.ToInt32(lstNumbers.Items[0]);
50                 for (int position = 0; position < lstNumbers.Items.Count;
                     position++)
```

```
51                  {
52                      if (Convert.ToInt32(lstNumbers.Items[position]) > maximum)
                         maximum = Convert.ToInt32(lstNumbers.Items[position]);
53                  }
54                  lblMaximumValue.Text = maximum.ToString();
55
56                  lblEntriesRemainingValue.Text = (maxnumberoflistitems -
                      lstNumbers.Items.Count).ToString();
57                  lblMaximumEntriesValue.Text = maxnumberoflistitems.ToString();
58              }
59
60              else
61              {
62                  lblCountValue.Text = lstNumbers.Items.Count.ToString();
63                  lblMinimumValue.Text = "-";
64                  lblMaximumValue.Text = "-";
65                  lblFirstValue.Text = "-";
66                  lblLastValue.Text = "-";
67                  lblEntriesRemainingValue.Text = maxnumberoflistitems.ToString
                      ();
68                  lblMaximumEntriesValue.Text = maxnumberoflistitems.ToString();
69              }
70
71          }
72
73          public void unsortedInsert(int number)
74          {
75              lstNumbers.Items.Add(number);
76          }
77
78          public void sortedInsert(int number)
79          {
80              int insertposition = 0;
81
82              for (int position = 0; position < lstNumbers.Items.Count; position
                  ++)
83              {
84                  if (Convert.ToInt32(lstNumbers.Items[position]) > number)
85                  {
86                      insertposition = position;
87                      break;
88                  }
89                  else insertposition = lstNumbers.Items.Count;
90              }
91
92              lstNumbers.Items.Add(0);
93
94              for (int position = lstNumbers.Items.Count - 1; position >= 0;
                  position--)
95              {
96                  if (position == insertposition)
97                  {
98                      lstNumbers.Items[position] = number;
99                      break;
100                 }
101                 else lstNumbers.Items[position] = lstNumbers.Items[position -
```

```
                    1];
102             }
103         }
104
105         public void deleteNumber(int index)
106         {
107             if (lstNumbers.Items.Count > 0)
108             {
109                 for (int position = index; position < lstNumbers.Items.Count -
                        1; position++)
110                 {
111                     lstNumbers.Items[position] = lstNumbers.Items[position +
                        1];
112                 }
113             }
114
115             lstNumbers.Items.RemoveAt(lstNumbers.Items.Count - 1);
116
117             updateStatus();
118
119             if (lstNumbers.Items.Count < maxnumberoflistitems)
120             {
121                 btnInitialise.Enabled = true;
122                 btnInsert.Enabled = true;
123             }
124
125             if (lstNumbers.Items.Count < 2 && optUnsorted.Checked == true)
                  btnShuffle.Enabled = false;
126
127             if (lstNumbers.Items.Count == 0)
128             {
129                 btnSearch.Enabled = false;
130                 btnClear.Enabled = false;
131                 grpSearch.Enabled = false;
132             }
133
134             lstNumbers.SelectedIndex = -1;
135
136             if (lstNumbers.SelectedIndex == -1)
137             {
138                 btnDelete.Enabled = false;
139                 picBin.Enabled = false;
140             }
141         }
142
143         private void frmMain_Load(object sender, EventArgs e)
144         {
145             picBin.AllowDrop = true;
146             updateStatus();
147         }
148
149         private void lstNumbers_MouseDown(object sender, MouseEventArgs e)
150         {
151             if (lstNumbers.Items.Count > 0 && lstNumbers.SelectedIndex != -1)
152             {
153                 btnDelete.Enabled = true;
```

```
154                 picBin.Enabled = true;
155                 lstNumbers.DoDragDrop(lstNumbers.SelectedIndex.ToString(),
                      DragDropEffects.Copy);
156             }
157         }
158
159         private void btnInitialise_Click(object sender, EventArgs e)
160         {
161             int itemcount = lstNumbers.Items.Count;
162
163             for (int position = 0; position < (maxnumberoflistitems -
                  itemcount); position++)
164             {
165                 int number;
166                 bool numberexists = false;
167
168                 do
169                 {
170                     number = random.Next(100 + 1);
171                     numberexists = checkNumberExists(number);
172                 } while (numberexists == true);
173
174                 if (optUnsorted.Checked == true) unsortedInsert(number);
175                 else if (optSorted.Checked == true) sortedInsert(number);
176
177             }
178
179             if (lstNumbers.Items.Count > 0)
180             {
181                 grpSearch.Enabled = true;
182                 btnSearch.Enabled = true;
183                 btnClear.Enabled = true;
184             }
185
186             if (lstNumbers.Items.Count > 1 && optUnsorted.Checked == true)
                  btnShuffle.Enabled = true;
187
188             if (lstNumbers.Items.Count == maxnumberoflistitems)
189             {
190                 btnInsert.Enabled = false;
191                 btnInitialise.Enabled = false;
192             }
193
194             updateStatus();
195         }
196
197         private void btnInsert_Click(object sender, EventArgs e)
198         {
199             int number;
200
201             try
202             {
203                 number = int.Parse(txtUserInput.Text);
204             }
205             catch (System.FormatException)
206             {
```

```
207                 MessageBox.Show("Input must be a number");
208                 txtUserInput.Text = "";
209                 txtUserInput.Focus();
210                 return;
211             }
212             catch (System.OverflowException)
213             {
214                 MessageBox.Show("Interger Overflow, Number must be in the
                      range of an unsigned integer (-2,147,483,648 to
                      2,147,483,647)");
215                 txtUserInput.Text = "";
216                 txtUserInput.Focus();
217                 return;
218             }
219
220             if (number < 0 || number > 100)
221             {
222                 MessageBox.Show("Number must be between 0 and 100 inclusive");
223                 txtUserInput.Text = "";
224                 txtUserInput.Focus();
225                 return;
226             }
227
228             if (checkNumberExists(number) == true)
229             {
230                 MessageBox.Show("Number must be unique");
231                 txtUserInput.Text = "";
232                 txtUserInput.Focus();
233                 return;
234             }
235
236             if (optUnsorted.Checked == true) unsortedInsert(number);
237             else if (optSorted.Checked == true) sortedInsert(number);
238
239             if (lstNumbers.Items.Count > 0)
240             {
241                 grpSearch.Enabled = true;
242                 btnSearch.Enabled = true;
243                 btnClear.Enabled = true;
244             }
245
246             if (lstNumbers.Items.Count > 1 && optUnsorted.Checked == true)
                  btnShuffle.Enabled = true;
247
248             if (lstNumbers.Items.Count == maxnumberoflistitems)
249             {
250                 btnInsert.Enabled = false;
251                 btnInitialise.Enabled = false;
252             }
253
254             txtUserInput.Text = "";
255             txtUserInput.Focus();
256
257             updateStatus();
258         }
259
```

```
260         private void btnShuffle_Click(object sender, EventArgs e)
261         {
262             int source, destination;
263 
264             for (int position = 0; position < lstNumbers.Items.Count; position
                ++)
265             {
266                 do
267                 {
268                     source = random.Next(lstNumbers.Items.Count);
269                     destination = random.Next(lstNumbers.Items.Count);
270                 } while (source == destination);
271 
272                 object temp = lstNumbers.Items[source];
273                 lstNumbers.Items[source] = lstNumbers.Items[destination];
274                 lstNumbers.Items[destination] = temp;
275             }
276 
277             updateStatus();
278         }
279 
280         private void btnSearch_Click(object sender, EventArgs e)
281         {
282             int number;
283 
284             try
285             {
286                 number = int.Parse(txtUserInput.Text);
287             }
288             catch (System.FormatException)
289             {
290                 MessageBox.Show("Input must be a number");
291                 txtUserInput.Text = "";
292                 txtUserInput.Focus();
293                 return;
294             }
295             catch (System.OverflowException)
296             {
297                 MessageBox.Show("Interger Overflow, Number must be in the
                    range of an unsigned integer (-2,147,483,648 to
                    2,147,483,647)");
298                 txtUserInput.Text = "";
299                 txtUserInput.Focus();
300                 return;
301             }
302 
303             if (number < 0 || number > 100)
304             {
305                 MessageBox.Show("Number must be between 0 and 100 inclusive");
306                 txtUserInput.Text = "";
307                 txtUserInput.Focus();
308                 return;
309             }
310 
311             int numberofprobes = 0;
312             bool numberfound = false;
```

```
313
314             if (optLinear.Checked == true)
315             {
316                 for (int position = 0; position < lstNumbers.Items.Count;
                      position++)
317                 {
318                     numberofprobes++;
319                     if (number == Convert.ToInt32(lstNumbers.Items[position]))
320                     {
321                         numberfound = true;
322                         MessageBox.Show("Search query: " + number.ToString() +
                         "\n Found: True \n List index: " + position.ToString() +
                        "\n Number of search probes : " + numberofprobes.ToString
                        ());
323                         break;
324                     }
325                     else numberfound = false;
326                 }
327
328                 if (numberfound == false) MessageBox.Show("Search query: " +
                      number.ToString() + "\n Found: False \n Number of search
                      probes : " + numberofprobes.ToString());
329             }
330
331             else if(optBinary.Checked == true)
332             {
333                 int searchbegin = 0;
334                 int searchend = lstNumbers.Items.Count - 1;
335                 int searchmid;
336
337                 while(!(searchbegin > searchend))
338                 {
339                     numberofprobes++;
340                     searchmid = (searchbegin + searchend) / 2;
341
342                     if (Convert.ToInt32(lstNumbers.Items[searchmid]) < number)
                         searchbegin = searchmid + 1;
343                     else if (Convert.ToInt32(lstNumbers.Items[searchmid]) >
                        number) searchend = searchmid - 1;
344                     else if (Convert.ToInt32(lstNumbers.Items[searchmid]) ==
                        number)
345                     {
346                         numberfound = true;
347                         MessageBox.Show("Search query: " + number.ToString() +
                         "\n Found: True \n List index: " + searchmid.ToString() +
                         "\n Number of search probes : " + numberofprobes.ToString
                        ());
348                         break;
349                     }
350                 }
351
352                 if (numberfound == false) MessageBox.Show("Search query: " +
                      number.ToString() + "\n Found: False \n Number of search
                      probes : " + numberofprobes.ToString());
353             }
354
```

```
355             txtUserInput.Text = "";
356             txtUserInput.Focus();
357         }
358
359         private void btnClear_Click(object sender, EventArgs e)
360         {
361             for (int position = lstNumbers.Items.Count - 1; position >=0;
                  position--)
362             {
363                 deleteNumber(position);
364             }
365         }
366
367         private void btnExit_Click(object sender, EventArgs e)
368         {
369             DialogResult buttonPressed;
370
371             buttonPressed = MessageBox.Show("Are you sure you want to exit?",
                  "Exit", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
372
373             if (buttonPressed == DialogResult.Yes) this.Close();
374         }
375
376         private void btnDelete_Click(object sender, EventArgs e)
377         {
378             deleteNumber(lstNumbers.SelectedIndex);
379         }
380
381         private void picBin_DragEnter(object sender, DragEventArgs e)
382         {
383             e.Effect = DragDropEffects.Copy;
384         }
385
386         private void picBin_DragDrop(object sender, DragEventArgs e)
387         {
388             int index;
389
390             try
391             {
392                 index = int.Parse(e.Data.GetData
                      (DataFormats.StringFormat).ToString());
393             }
394             catch
395             {
396                 MessageBox.Show("Input must be a number");
397                 return;
398             }
399
400             if (index < 0 || index >= lstNumbers.Items.Count)
401             {
402                 MessageBox.Show("Input must be a valid list index");
403                 return;
404             }
405
406             deleteNumber(index);
407         }
```

```
408
409         private void optUnsorted_Click(object sender, EventArgs e)
410         {
411             if (lstNumbers.Items.Count > 1) btnShuffle.Enabled = true;
412             optBinary.Enabled = false;
413             optLinear.Select();
414         }
415
416         private void optSorted_Click(object sender, EventArgs e)
417         {
418             object temp;
419             bool swap;
420             do
421             {
422                 swap = false;
423
424                 for (int position = 0; position < lstNumbers.Items.Count - 1;
                      position++)
425                 {
426                     if (Convert.ToInt32(lstNumbers.Items[position]) >
                        Convert.ToInt32(lstNumbers.Items[position + 1]))
427                     {
428                         temp = lstNumbers.Items[position];
429                         lstNumbers.Items[position] = lstNumbers.Items[position
                         + 1];
430                         lstNumbers.Items[position + 1] = temp;
431                         swap = true;
432                     }
433                 }
434             } while (swap);
435
436             optBinary.Enabled = true;
437             btnShuffle.Enabled = false;
438
439             updateStatus();
440         }
441     }
442 }
443
```