

Event Driven And Graphical User Interface Programming

CS2S564\_2018\_v1

Connor Haines | 17025931 | Due 14/12/2018 11:59

# Design

I will go through my design from the top of the form downwards to explain my decisions.

### Form Design

To begin, I named the form Number Generator & Sorter as I felt that that covers the basic premise of what this form does. I included an .ico file which shows a generator symbol just to link back to the name of the form. I decided the right size for **my** form would be 347 pixels wide and 441 pixels down. I also made sure that the form could not be made smaller or larger.

### List Box

My list box was made to contain the numbers that the users insert (or initialize) into the list and so users could make the choice to see the list in either an unsorted list or a sorted list. During the design stages I named this list box lst\_Numbers. I used the lst shorthand to let any other people that may look into the code know that it was a listbox and I used the word Numbers as then it is quite obvious that it is a list of numbers.

### Search Group Box

Inside my Search Group Box, I have put two radio buttons that control what type of search will run when the search button (also in the group box) is clicked. The first radio button is for Linear search and the other for Binary and are named as such. The Group Box also includes the text box for users to type a number that they want to search for. The names of the radio buttons are rad\_Binary and rad\_Linear. The search text box and button are named tb\_Search and btn\_Search

### Drag N’ Drop Bin Group Box

Within this list box is only one thing and that is the Icon of a trash can with Drag and Drop functionality. The reason I put this into a group box is because it matched the rest of the design and I find consistency is a big thing within a GUI. It is named pb\_Bin as it is a picture box and the image is of a trash can.

### Insert Group Box

The Insert Group Box includes two buttons called btn\_Insert and btn\_Initialize. Both are used to put numbers into the list box. It also includes two radio buttons that decide how the list will be sorted these are called rad\_Sort and rad\_Unsort. The final thing in the Group Box is text box that allows the users to input an integer into the list box when the insert button is clicked. This is named tb\_Insert.

### Statistics Group Box

My statistics Group Box is comprised of 5 labels that get updated throughout the form. The first label is Number of Entries, this shows how many values are actually in the form. The next label is the First Value, this shows what value is at the top of the list.   
Third is Last Value, this shows the value at the bottom of the list.  
Second to last is Minimum, which finds the smallest value and shows it.  
Finally is Maximum, this finds the largest value and shows it.

### Controls Group Box

The Controls Group Box contains four buttons that control the list box in different ways. The delete button, named btn\_Delete, is used to delete the number that is selected in the list box if there is more than one number. The next button is the shuffle button, also named appropriately btn\_Shuffle, this is only enabled if there is more than two numbers and the unsorted radio button is checked. The clear button is used to clear the list. This is called btn\_Clear. Finally, the exit button is used to close the form.

## Pseudocode

This will be used to show a few examples of pseudocode used within the program.

### Linear Search

If (linear search button is selected)  
{  
 For (the count of the list )  
 {  
 If (search value = value in list)  
 {  
 Return value of index  
 Select index  
 }  
 }  
}

### Duplicate Function

For (the count of the list)  
 {  
 If (an integer that we put in the parameters == a   
 number in the list)  
 {  
 Return 1 so that it can be used in an ‘if’ later

} otherwise return 0

### Initialize Button

While (count of list < max amount of numbers allowed)  
{  
 create random number  
 If (Random number is not in the list)  
 {  
 Insert (random number)  
 }  
}

### Linear Search Function

Make a new variable with the value of whatever the user   
 searched for

Make a new int with the starting value of 0

For (count of list)  
 {  
 If( number found matches number searched for)  
 {  
 show the number in list   
 return a number to be used in a later ‘if’  
 } else return different number  
 }

# Implementation

For the readers ease of use, I have changed the colour of the text that were comments so that it does not get in your way or obscure the implementation code when reading but is also there for if you need extra explanation.

//17025931

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 coursework

{

public partial class Form1 : Form

{

const int MAXNUMBERS = 30; //declare const int MAXNUMBERS as 30 so it can be used as a global variable

int searchNumber = 0; //declare searchNumber as a global variable

int mid = 0; //declare mid as a global variable

int tracker = 0; //declare tracker as a global variable

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

Update\_Stats(); //calls update stats

pb\_Bin.AllowDrop = true; //sets allowdrop for the picture box d&d

}

private int DupFun(int integer)

{

for (int i = 0; i < lst\_Numbers.Items.Count; i++) //loops through the lists item count

{

if (integer == int.Parse(lst\_Numbers.Items[i].ToString())) //until it finds a duplicate

{

return 1; //if it finds one it returns 1

}

}

return 0; //if it doesn't it returns 0

}

private void bubble\_sort()

{

int t; //declare int t

for (int p = 0; p <= lst\_Numbers.Items.Count - 2; p++) //while p is less than or equal to the count of the list(minus2) it will loop

{

for (int i = 0; i <= lst\_Numbers.Items.Count - 2; i++) //loops through the lists item count

{

if (int.Parse(lst\_Numbers.Items[i].ToString()) > int.Parse(lst\_Numbers.Items[i + 1].ToString())) //if i is greater than i+1

{

t = int.Parse(lst\_Numbers.Items[i + 1].ToString()); //change the value of t

lst\_Numbers.Items[i + 1] = lst\_Numbers.Items[i]; //make item[i] equal to item[i + 1]

lst\_Numbers.Items[i] = t; //makes item[i] equal to t

}

}

}

}

private void btn\_Exit\_Click(object sender, EventArgs e)

{

this.Close(); //closes the form

}

private void btn\_Shuffle\_Click(object sender, EventArgs e)

{

ListBox.ObjectCollection list = lst\_Numbers.Items; //Represents the collection of items in a ListBox.

Random rng = new Random(); //create new random

int i = list.Count; //declare i and make it equal to the amount of values in the list

lst\_Numbers.BeginUpdate(); //prevents the control from drawing until end update is called

while (i > 1) //while the number of values in the list is more than one

{

i--; //take one away

int n = rng.Next(i + 1); //create n and make it equal to the next random (i + 1)

object value = list[n]; //makes value equal to list[n]

list[n] = list[i]; //makes list[n] equal to list [i]

list[i] = value; //makes list[i] equal to value

}

lst\_Numbers.EndUpdate();

lst\_Numbers.Invalidate();

Update\_Stats(); //calls update stats

}

private void btn\_Initialise\_Click(object sender, EventArgs e)

{

lst\_Numbers.Items.Clear(); //clears the listbox

if (rad\_Unsort.Checked == true) //if the unsorted radio button is checked

{

var rand = new Random();

//for (int i = 0; i < 30; i++)

while (lst\_Numbers.Items.Count < MAXNUMBERS) //loops until list count is equal to 30

{

int index = rand.Next(0, 101); //random number between 0 and 101

if (DupFun(index) != 1) //if duplicate function returns anything except a 1

lst\_Numbers.Items.Add(index); //add to the end

}

}

else if (rad\_Sort.Checked == true)

{

var rand = new Random();

//for (int i = 0; i < 30; i++)

while (lst\_Numbers.Items.Count < MAXNUMBERS)

{

int index = rand.Next(0, 101);

if (DupFun(index) != 1)

lst\_Numbers.Items.Add(index);

}

bubble\_sort(); //calls bubble sort

}

Update\_Stats(); //calls update stats

}

private int insertSorted()

{

for (int i = 0; i < lst\_Numbers.Items.Count; i++) //loops through the list count

{

if (int.Parse(tb\_Insert.Text) < int.Parse(lst\_Numbers.Items[i].ToString())) //if the number inserted is less than the number we're looking at

{

lst\_Numbers.Items.Insert(i, ""); //inserts blank

lst\_Numbers.Items[i] = int.Parse(tb\_Insert.Text); //set value of blank to the number user inserted

return 1; //return 1

}

}

return 0; //otherwise return 0

}

private void btn\_Insert\_Click(object sender, EventArgs e)

{

int test;//check for int datatype

if (tb\_Insert.Text != "") //check they don't click insert straight away

{

if (!string.IsNullOrEmpty(tb\_Insert.Text)) //check the string isn't null or empty

{

if (int.TryParse(tb\_Insert.Text, out test)) //tries to parse it to an integer

{

if (DupFun(int.Parse(tb\_Insert.Text.ToString())) != 1) //if duplicate function doesn't return 1

{

if (int.Parse(tb\_Insert.Text) >= 0 && int.Parse(tb\_Insert.Text) <= 100) //if the number is less than 100 and over 0 (or equal to)

{

if (rad\_Sort.Checked == true) //if the sort radio button is checked

{

if (insertSorted() == 0) //and insert sorted returns a 0

{

lst\_Numbers.Items.Add(int.Parse(tb\_Insert.Text)); //add to the end

}

}

if (rad\_Unsort.Checked == true) //if the unsort radio button is checked

{

lst\_Numbers.Items.Add(int.Parse(tb\_Insert.Text)); //add to the end

}

}

else MessageBox.Show("Please insert values between 0 and 100!");

}

else MessageBox.Show("This number has already been inserted!");

}

else MessageBox.Show("Please insert an integer only!");

}

else MessageBox.Show("Please insert a value!");

}

else MessageBox.Show("Please insert a value!");

Update\_Stats(); //call update stats

}

private void Update\_Stats() //this whole thing updates stats

{

if (lst\_Numbers.Items.Count == 0)

{

lbl\_NumberEntries.Text = "Number of Entries: 0 ";

lbl\_First.Text = "First Value: ";

lbl\_Last.Text = "Last Value: ";

lbl\_Max.Text = "Maximum: ";

lbl\_Min.Text = "Minimum: ";

tb\_Insert.Text = String.Empty;

btn\_Search.Enabled = false;

tb\_Search.Enabled = false;

btn\_Clear.Enabled = false;

btn\_Delete.Enabled = false;

}

else

{

lbl\_NumberEntries.Text = "Number of Entries: " + lst\_Numbers.Items.Count;

lbl\_First.Text = "First Value: " + lst\_Numbers.Items[0].ToString();

lbl\_Last.Text = "Last Value: " + lst\_Numbers.Items[lst\_Numbers.Items.Count - 1].ToString();

int maxNumber;

int minNumber;

maxNumber = int.Parse(lst\_Numbers.Items[0].ToString());

minNumber = int.Parse(lst\_Numbers.Items[0].ToString());

for (int i = 0; i < lst\_Numbers.Items.Count; i++)

{

int currentNumber = int.Parse(lst\_Numbers.Items[i].ToString());

if (currentNumber > maxNumber)

{

maxNumber = currentNumber;

}

if (currentNumber < minNumber)

{

minNumber = currentNumber;

}

}

lbl\_Max.Text = "Maximum: " + maxNumber;

lbl\_Min.Text = "Minimum: " + minNumber;

tb\_Insert.Text = String.Empty;

tb\_Search.Text = String.Empty;

btn\_Search.Enabled = true;

tb\_Search.Enabled = true;

btn\_Clear.Enabled = true;

btn\_Delete.Enabled = true;

}

if (rad\_Unsort.Checked == true)

{

btn\_Shuffle.Enabled = true;

rad\_Binary.Enabled = false;

rad\_Linear.Checked = true;

}

else if (rad\_Sort.Checked == true)

{

rad\_Binary.Enabled = true;

btn\_Shuffle.Enabled = false;

}

if (lst\_Numbers.Items.Count == MAXNUMBERS)

{

btn\_Insert.Enabled = false;

tb\_Insert.Enabled = false;

btn\_Initialise.Enabled = false;

}

else

{

btn\_Insert.Enabled = true;

tb\_Insert.Enabled = true;

btn\_Initialise.Enabled = true;

}

if (lst\_Numbers.Items.Count >= 2)

{

btn\_Shuffle.Enabled = true;

}

else

{

btn\_Shuffle.Enabled = false;

}

if (lst\_Numbers.Items.Count > 0 && rad\_Sort.Checked == true )

{

btn\_Shuffle.Enabled = false;

}

else if(lst\_Numbers.Items.Count < 30 && rad\_Sort.Checked == true && lst\_Numbers.Items.Count != 0)

{

btn\_Shuffle.Enabled = true;

}

}

private int linSearch()

{

searchNumber = int.Parse(tb\_Search.Text); //make a new variable with the value of whatever the user searched for

int Found = 0; //make a new int with the starting value of 0

for (int i = 0; i < lst\_Numbers.Items.Count; i++) //loop through the list

{

if (lst\_Numbers.Items[i].ToString() == searchNumber.ToString()) //if the number found matches the number that the user searched for

{

lst\_Numbers.SetSelected(i, true); //set the number to selected

Found = 1; //set found to 1

return i; //return i

}

}

if (Found == 0) //if it is not found

{

return -1; //it will return -1

}

return 0; //otherwise will return 0

}

private int binSearch()

{

int min = 0; //make variable min equal to 0

int max = lst\_Numbers.Items.Count - 1; //make variable max equal to the number of items in the list - 1

while (min <= max) //loop while min is less than or equal to max

{

mid = (min + max) / 2; //change the value of mid to equal min + max divided by two

tracker++; //add 1 to the tracker variable

if (int.Parse(tb\_Search.Text) == int.Parse(lst\_Numbers.Items[mid].ToString())) //if the number the user searched for equals the number found

{

return mid; //return mid

}

else if (int.Parse(tb\_Search.Text) < int.Parse(lst\_Numbers.Items[mid].ToString())) //if the number the user searched for is less than the number found

{

max = mid - 1; //changes the max to mid - 1

}

else //otherwise

{

min = mid + 1; //change min to mid + 1

}

}

return -1; //returns -1

}

private void btn\_Search\_Click(object sender, EventArgs e)

{

int test;//check for int datatype

if (tb\_Search.Text != "") //checks that they didn't click search immediately

{

if (!string.IsNullOrEmpty(tb\_Search.Text)) //checks its not null or empty

{

if (int.TryParse(tb\_Search.Text, out test)) //tries to parse it as an int

{

if (int.Parse(tb\_Search.Text) >= 0 && int.Parse(tb\_Search.Text) <= 100) //checks the searchnumber is between 1 and 100 or equal to

{

if (rad\_Linear.Checked == true) //if the linear button is checked

{

if (linSearch() == -1) //if the linear search returned -1

{

MessageBox.Show("Sorry! We could not find your number! Please either try again or search for a different number.");

}

else //otherwise..

{

MessageBox.Show(searchNumber + " is in the list as position " + (linSearch() + 1) + ". Found with Linear Search");

MessageBox.Show(" We did " + (linSearch() + 1) + " comparisons.");

}

}

if (rad\_Binary.Checked == true) //if the binary button is checked

{

if (binSearch() == -1) //if the binary search returned -1

{

MessageBox.Show("Sorry! We could not find your number! Please either try again or search for a different number.");

}

else //otherwise..

{

lst\_Numbers.SetSelected(mid, true); //selects the number and highlights it

MessageBox.Show(tb\_Search.Text + " is in the list at position " + (mid+1) + ". Found with Binary Search.");

MessageBox.Show(" We did " + tracker + " comparisons.");

}

}

}

else MessageBox.Show("Please insert values between 0 and 100!");

}

else MessageBox.Show("Please insert an integer only!");

}

else MessageBox.Show("Please insert a value!");

}

else MessageBox.Show("Please insert a value!");

Update\_Stats();

}

private void btn\_Clear\_Click(object sender, EventArgs e)

{

lst\_Numbers.Items.Clear(); //clears the list

Update\_Stats(); //calls update stats

}

private void Delete()

{

ListBox.SelectedObjectCollection selectedItems = new ListBox.SelectedObjectCollection(lst\_Numbers); //sets selecteditems to a new selected object using lst\_numbers

selectedItems = lst\_Numbers.SelectedItems; //makes selecteditems equal to the list number selected items

if (lst\_Numbers.SelectedIndex != -1) //if the selected index does not equal -1

{

for (int i = selectedItems.Count - 1; i >= 0; i--) //loops until there is nothing left in the list

lst\_Numbers.Items.Remove(selectedItems[i]); //removes selected item

}

else

MessageBox.Show("Select a number to delete! ");

}

private void btn\_Delete\_Click(object sender, EventArgs e)

{

Delete(); //deletes whatever is selecter

Update\_Stats(); //calls update stats

}

private void btn\_Sort\_CheckedChanged(object sender, EventArgs e)

{

bubble\_sort(); //if the sort buttons check changes it will call bubblesort..

Update\_Stats(); //..update stats..

rad\_Binary.Enabled = true ; //..and will enable the binary button..

btn\_Shuffle.Enabled = false; //..and disable the shuffle button

}

private void pb\_Bin\_DragDrop(object sender, DragEventArgs e)

{

Delete(); //deletes whatever number was dropped on top of it

Update\_Stats(); // calls update stats

}

private void lst\_Numbers\_MouseDown\_1(object sender, MouseEventArgs e)

{

lst\_Numbers.DoDragDrop(lst\_Numbers.SelectedIndex, DragDropEffects.Move); //starts the drag drop when you start dragging a number

}

private void pb\_Bin\_DragEnter\_1(object sender, DragEventArgs e)

{

e.Effect = DragDropEffects.Move; //moves the number

}

private void btn\_Unsort\_CheckedChanged(object sender, EventArgs e)

{

Update\_Stats(); //if the unsort buttons check changes it will call update stats

}

}

} //430 lines of pure blood, sweat and tears.

# Testing

All Screenshots will be below the table and will have the correlating number as a header.

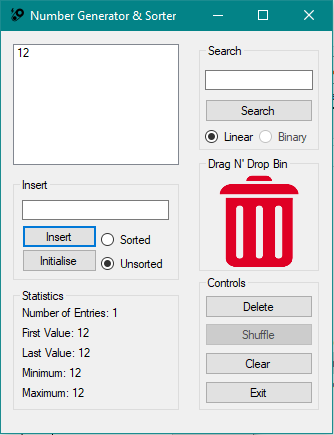
|  |  |  |  |
| --- | --- | --- | --- |
| Test Number | Test | Expected Result | Actual Result |
| 1 | I will run the form to make sure that it compiles and runs as expected. | I expect that the form will load up with the list box empty and the linear search button will be selected, and the unsorted button will be selected. | The form loaded up with the list box empty and the linear search button was selected, and the unsorted button was selected. |
| 2 | I will try to click the insert number without inserting a value to be inserted. | I should be met with a message box telling me that I must input a value. | I was met with a message box telling me that I must input a value. |
| 3 | I will try to insert the number 12 whilst the unsorted radio button is checked. | The number 12 should be added to the list box at the end, as this is the first number it should be the only number in the list. | The number 12 was added to the list and was the only number in it. |
| 4 | I will attempt to shuffle the list with only one value in the list. | As there is only one number in the list the shuffle button should be disabled, therefore not allowing me to shuffle the list. | The shuffle button was disabled and therefore I could not use it. |
| 5 | I will use the initialize button whilst there is already a number in the list and the radio button unsorted is checked. | As per the lecturer, we do not want the list box to be ‘topped up’ therefore I expect the list box to be cleared and then filled with 30 unique random numbers ranging from 1-100. | The list box was cleared, and 30 unique numbers were added to the list in an unsorted fashion. |
| 6 | I will use the clear button to clear the list box of all values. | The list box should be empty once the button is clicked. | After the button was clicked the list box was empty. |
| 7 | I am testing to see if the delete button is disabled when you clear the list | When the clear button is clicked then the delete button should be disabled. | The delete button was disabled. |
| 8 | I am testing to see if the clear button gets disabled after you have clicked clear already. | Once the clear button is clicked, it should immediately be disabled until there is 1 or more value back in the list. | After clicking the clear button once, it disabled itself. |
| 9 | I will now try to initialize the list whilst the sorted radio button is checked. | The list box should be filled with 30 unique random numbers and sorted from lowest to highest. | The list box was filled with 30 unique random numbers and sorted from lowest to highest. |
| 10 | I will now try to use binary search to search for the number 2. Whilst the sorted radio button is checked. | If the number is in the list box then we should get a messagebox stating where in the list it is and how many comparisons we had to do to find it. It will also be selected. Otherwise, it should tell us that it wasn’t found. | As the number was found in the test, a messagebox popped up telling us where in the list it was found and how many comparisons it took to find it. To make it easier, the number was also highlighted in the list. |
| 11 | I will try to use binary search to search for the number 13 whilst the unsorted radio button is checked. | Binary search should only be used when the sorted button is checked. Therefore, I expect the linear radio button to become checked and the binary radio button to become disabled. | The linear radio button became checked and the binary radio button got disabled. |
| 12 | I am going to try to insert the number 2 in to the list box twice when the unsorted radio button is checked. | The number should not be allowed to be added and we should get a messagebox popping up telling us that it is already in the list from the first time we inserted it. | A messagebox shows us that the number is already in the list and so the second number 2 will not be added to the list. |
| 13 | I will now try to insert numbers into the list whilst the sorted radio button is checked. | If I insert the numbers 69 and then 42. 42 should be placed in the list above 69. | 42 was inserted into the list above 69. |
| 14 | I will try to insert a number that is above or below my stated range. | I tried to insert the number -1. A messagebox should pop up and remind me of the specified range. | A messagebox popped up saying “Please insert a value between 1-100!”. |
| 15 | I will try to insert a string. | I tried to insert the word “string”. A messagebox should show up telling us that we can only insert integers. | A messagebox popped up saying “Please insert a full integer only”. |
| 16 | I will try to put in a number with zeros before it. | Hopefully, if somebody accidentally puts zeros before a number when they insert, like 05, it will ignore the zeros and insert the number. | It ignored the zeros and inserted the number without the zeros. |
| 17 | I will attempt to delete a number from a sorted list. | I tried to delete the number 12 from a full sorted list, I expect the number to be removed from the list and the remaining numbers will be moved up 1 in the position. | The number was removed, and the remaining numbers were moved up the list in position. |
| 18 | I will attempt to delete a number from the list using the drag and drop bin. | Once I’ve dragged the number over to the bin and let go the number should be removed from the list. | The number was removed from the list after dragging and dropping it on to the bin image. |
| 19 | I will test to see if the Statistics Group Box is working as it should. | When most things happen in the form the Update\_Stats() function is called and will update to fit into what is happening in the code. If I initialize a sorted list then I should see the first and last values are the same as the minimum and maximum. | When I initialized a sorted list, I saw that the first and last values were the same as the minimum and maximum values in the statistics section as you could plainly see in the list the statistics were showing the right values. |
| 20 | I am testing to see if the form can be resized. | I chose to not let my form be resized as I liked the way it looked and the size of it as is. Therefore, I expect it to not be able to be resized. | It was not able to be resized, you are not even given the option. |
| 21 | I am testing to see if the exit button works. | The exit button when clicked should immediately close the form, I made it this way as I felt the button was far enough out of the way that no one would click it accidentally. | Once the exit button was clicked, the form immediately closed. |
| 22 | I want to test if the list can be sorted after the numbers are inserted via the unsorted function. | I expect the form to update the list and set it out as if the numbers were initialized (or inserted) in the sorted manner. | The list did exactly what I expected and updated itself so that the numbers were sorted into the order of lowest to highest. |

# Screenshots

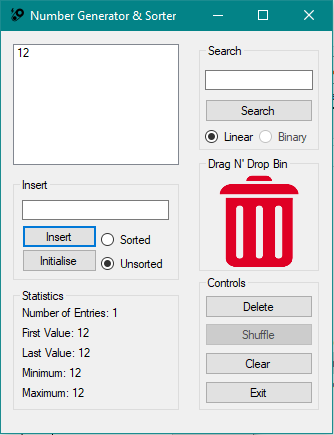
#### 1.

#### 2.

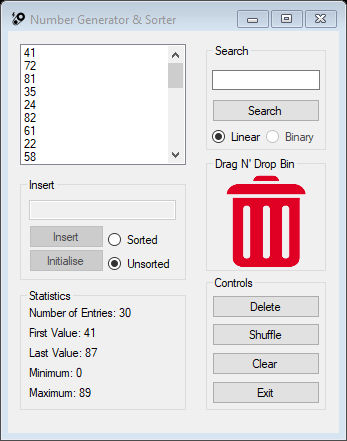
#### 3.



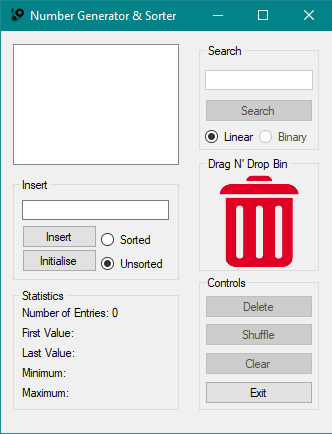
#### 4.



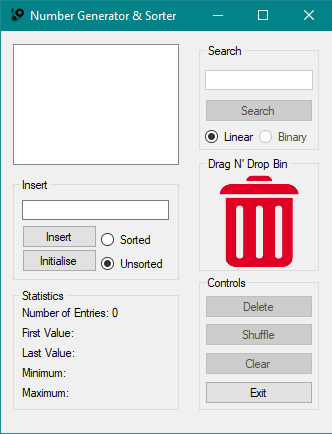
#### 5.



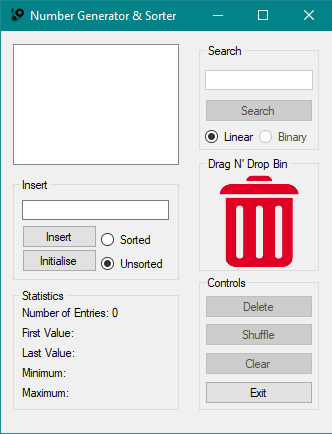
#### 6.



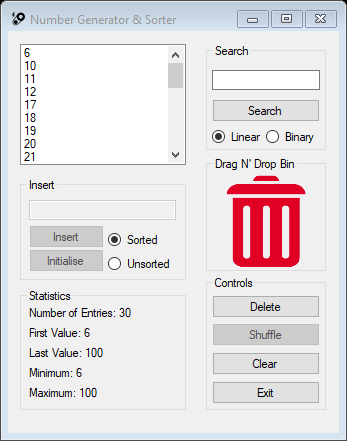
#### 7.



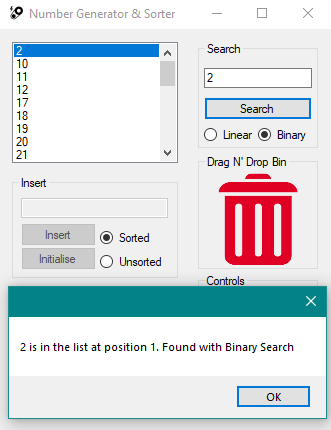
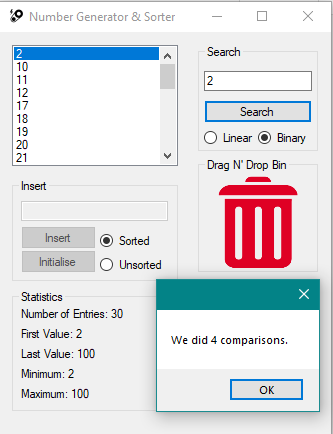
#### 8.



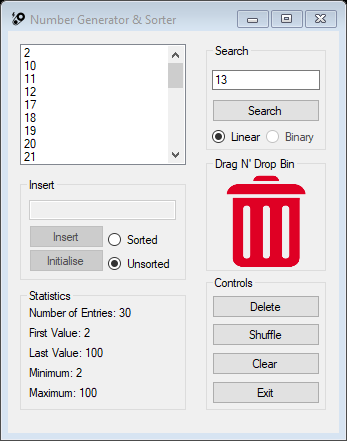
#### 9.



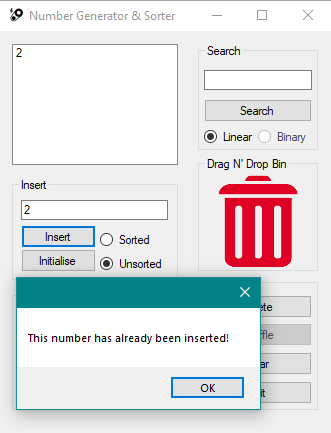
#### 10.



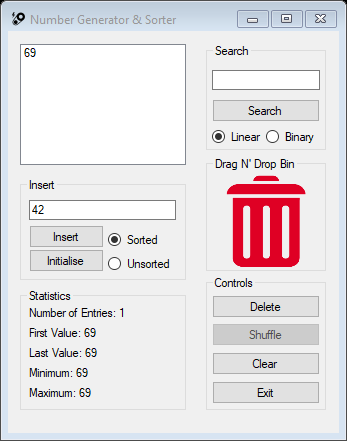
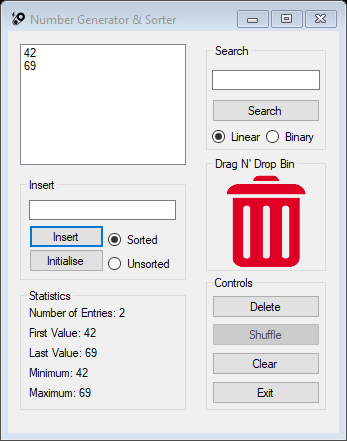
#### 11.



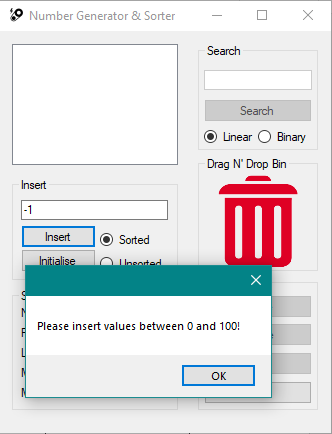
#### 12.



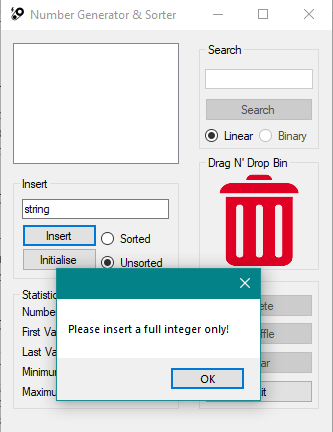
#### 13.



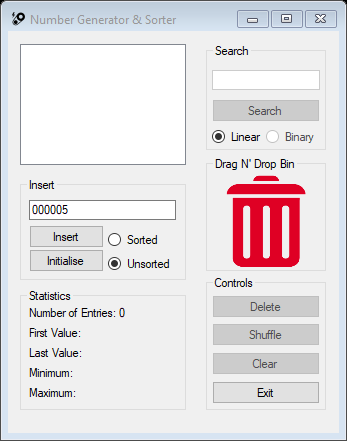
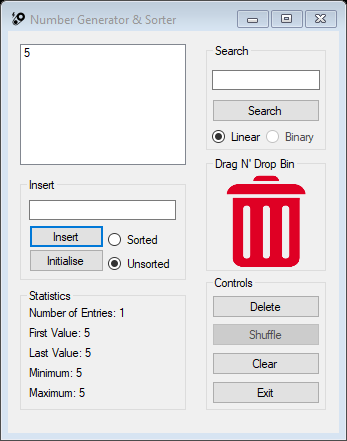
#### 14.



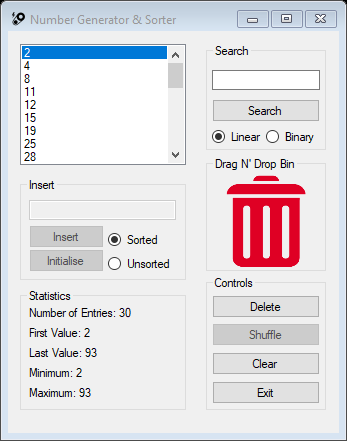
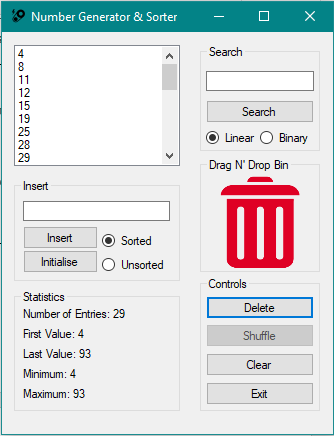
#### 15.



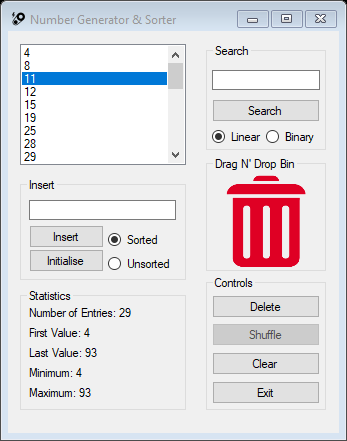
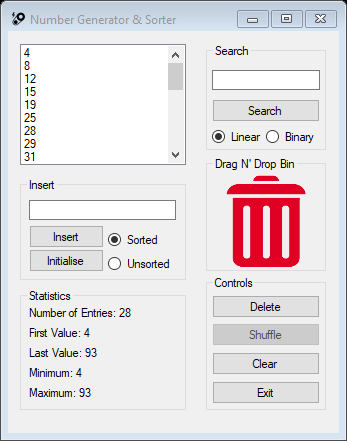
#### 16.



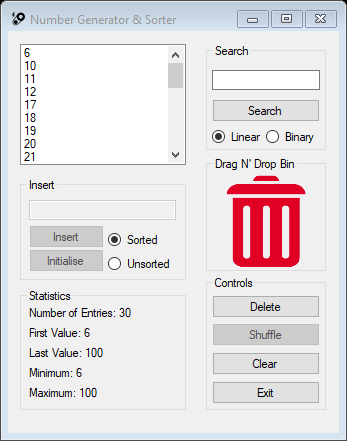
#### 17.



#### 18.



#### 19.



#### 20.



#### 21.



#### 22.

