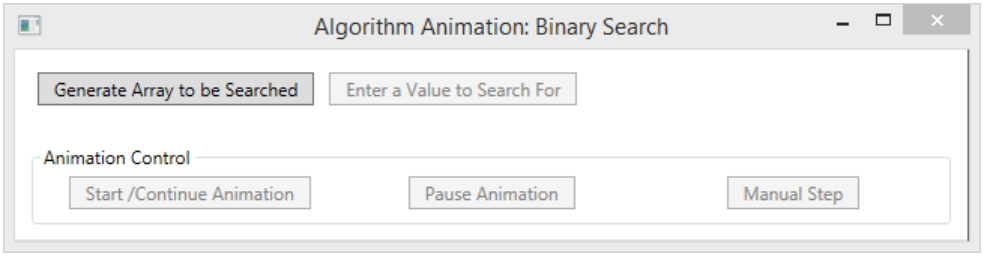
CSC 355 PROJECT 3
ALGORITHM ANIMATION: BINARY SEARCH

PROFESSOR GODFREY C. MUGANDA

Your assignment is to write a WPF program that animates the working of the Binary Search Algorithm.

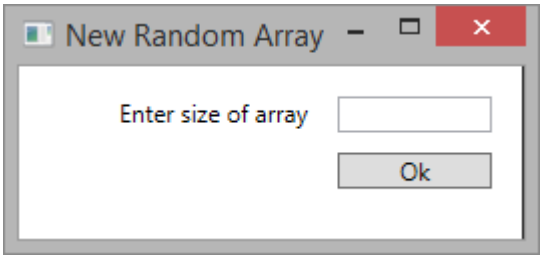
1. The User Interface

When the program is started, it displays the user interface shown below:

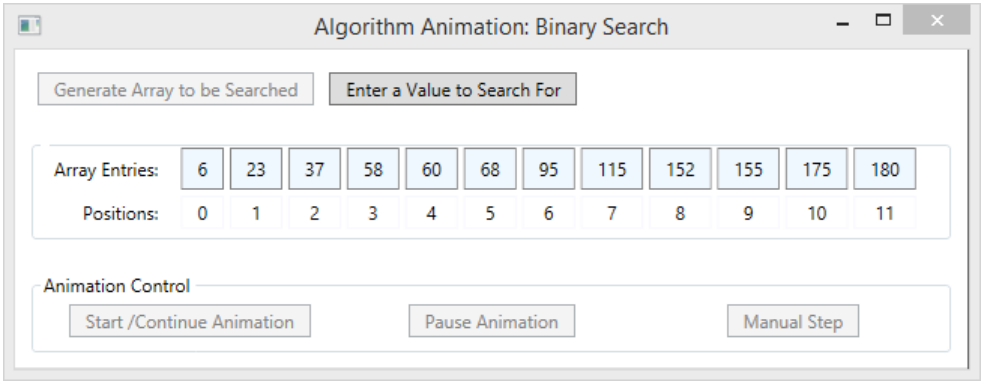


On this initial screen, only the button for generating an array to be searched is enabled.

On the initial screen, clicking the button to generate an array brings up a modal dialog box that solicits the size of the array to be generated:

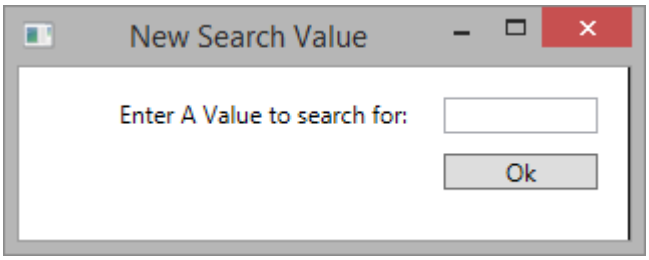


Upon the user entering a size and dismissing the dialog, the program adds to the user interface a `GroupBox` that contains a grid displaying a randomly generated array that has already been sorted. Array values should be in the range $0, \ldots, 200$. Note also that the indices indicating the position of a value within the array are shown.

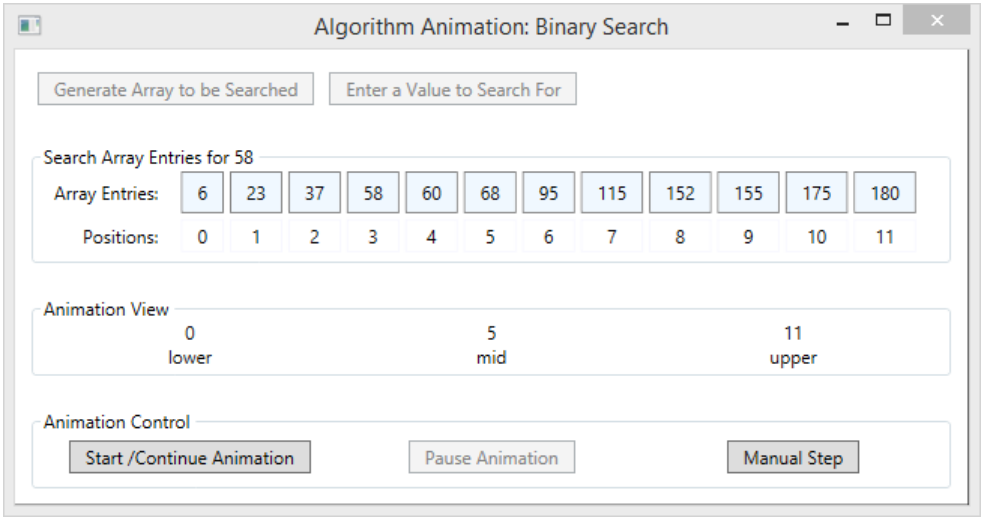


On this screen, the button to enter a value to search for is enabled, but no other buttons are enabled.

Upon the user clicking the button to specify a value to search for, the program displays another modal dialog for that purpose:



When the user enters a search value (say, 58) and dismisses this dialog, a header is added to the `GroupBox` displaying the array that identifies the value to search for, and a new group box, with the header `Animation View`, is added to the user interface.



Notice that the animation view shows the current values of the variables `lower`, `mid` and `upper` that play such an important role in the operation of *Binary Search*.

NOTE: As an option, you may combine the *Search Array Entries* GroupBox and the *Animation View* GroupBox into a single panel and display both as soon as

the array is generated. You can then add the header for the *Search Array Entries* GroupBox after the search value has been specified.

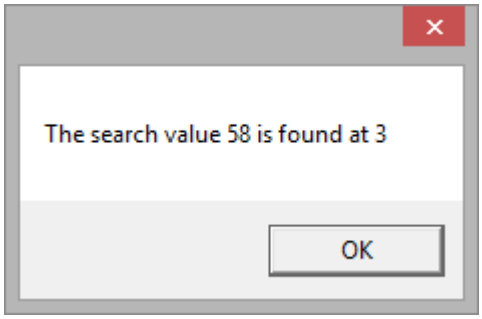
At this point, the Start Animation *and* `Manual Step` *buttons in the* Animation Control *group box should be enabled.*

2. How the Animation Operates

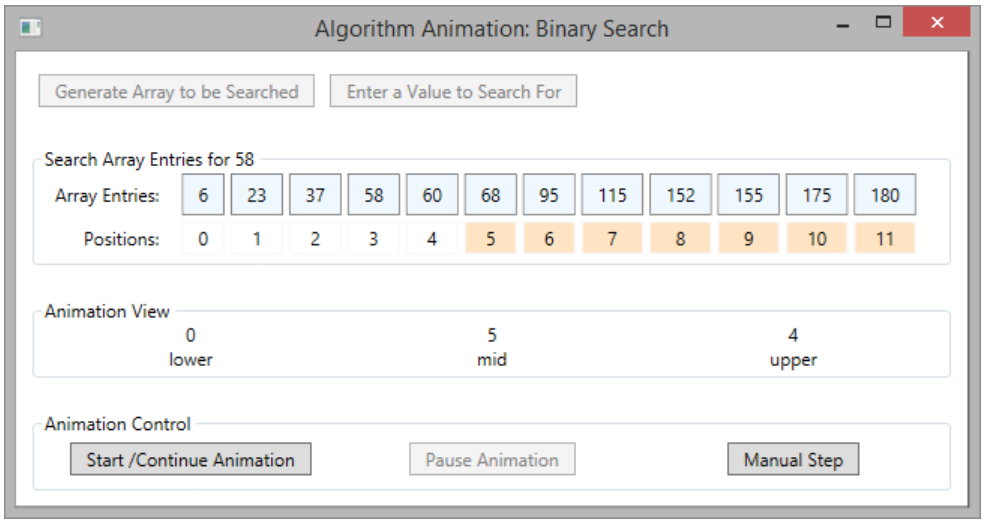
The animation can be operated manually be the user, by pressing the *Manual Step* button. Each press of the this button will run through a single iteration of binary search, by

(1) Displaying a message that the search value was not found if *lower* is greater than *upper* and disabling all buttons.

(2) Displaying a message that the value was found at *mid* (if that is indeed the case) and disabling all buttons.

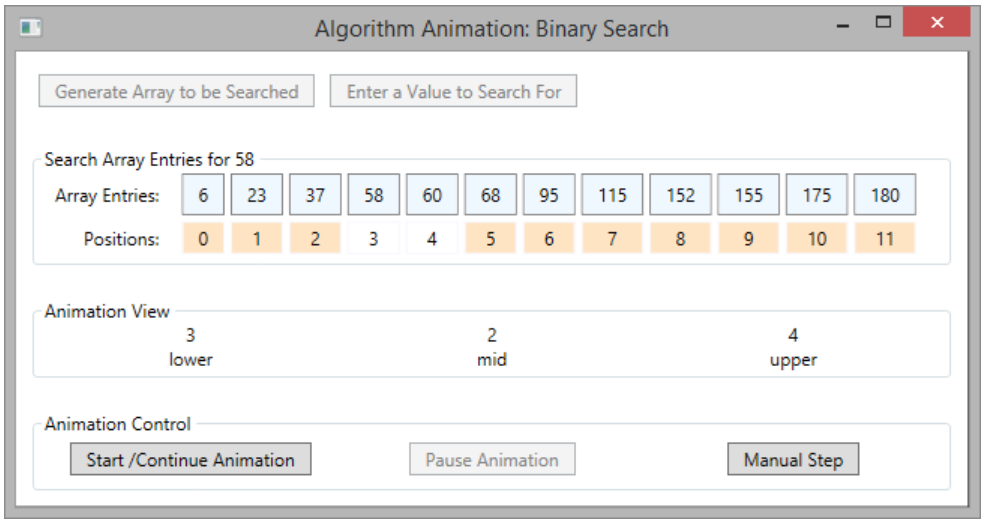


(3) Coloring the background of all positions that have been determined not to contain the search value and updating one of *lower* or *upper*.



For example, here the array values in positions 5 . . . 11 have been ruled out as being too large to contain the search value, 58.

In the screen shot below, the values in positions 0 . . . 2 have also been ruled out as being too small



The animation can also be operated on a time. Set a timer to generate a tick at one-second intervals, and at each tick, perform a single manual step.

The timer is started by pressing the *Start Animation* button, and the timer is paused, or stopped, py pressing the *Pause Animation* button.

The Manual Step *button should be disabled whenever the timer is running, but it should be enabled whenever the timer is not running.*

3. Due Date

Monday of Week 5.