Project 4

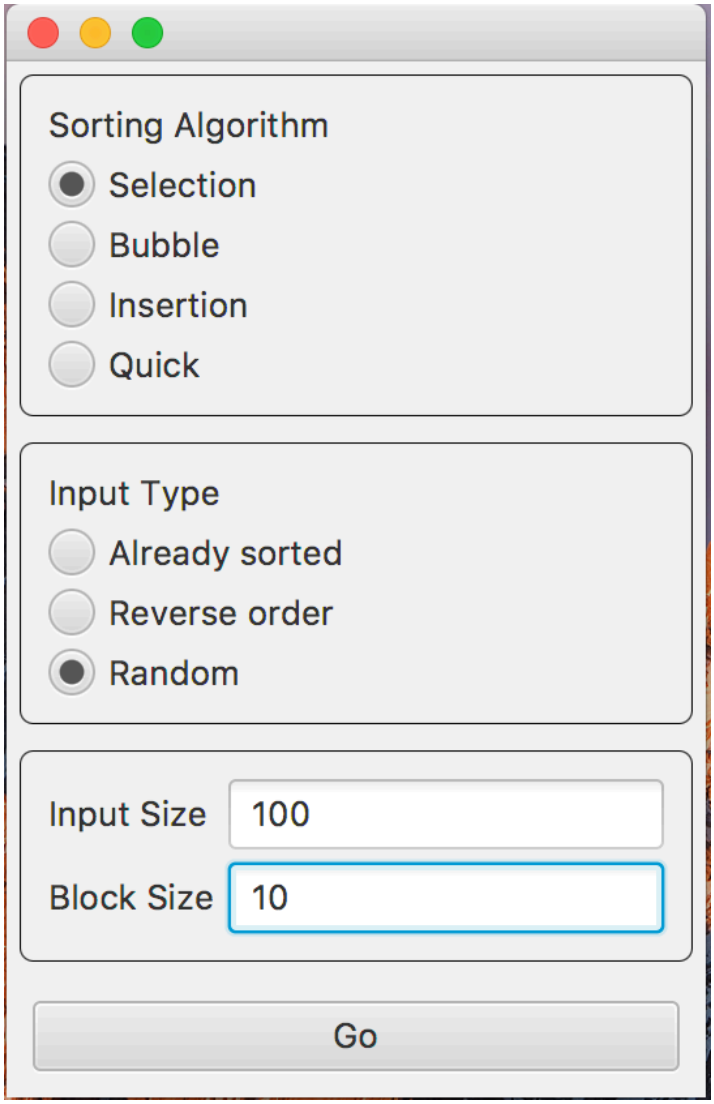
Your task is to write a program that allows the user to experiment with different multi-threaded sort algorithms. You should provide the user with an interface that looks very similar to the one shown below.

The user should be able to select the type of sorting algorithm to be used (selection, bubble, insertion, or quick), the size of the array to sort, the type of values to put in the array (already sorted in ascending order, sorted in reverse (descending) order, or random numbers between 0 and 99), and the number of values to sort in each thread.

When the user clicks on the Go button, you should validate all of the input. If the user has not provided required input or if the input is invalid (a negative input size, for example), you should display an alert dialog explaining the problem.

Otherwise, collect all of the input and generate an appropriate array of integer values. Then, "chunk" the array into the desired block size and pass each chunk to a thread that sorts the chunk into ascending order using the algorithm chosen by the user. When all of the chunks have been sorted, put the resulting sorted integer arrays into a queue. Then, poll two sorted chunks at a time from the queue, spawn a thread to merge them, and push the new merged array back onto the queue. Repeat this process until there is only a single integer array in the queue. This array should now be sorted. Display the sorted array to the console, and use an alert dialog to show the user how many milliseconds the sort process took.

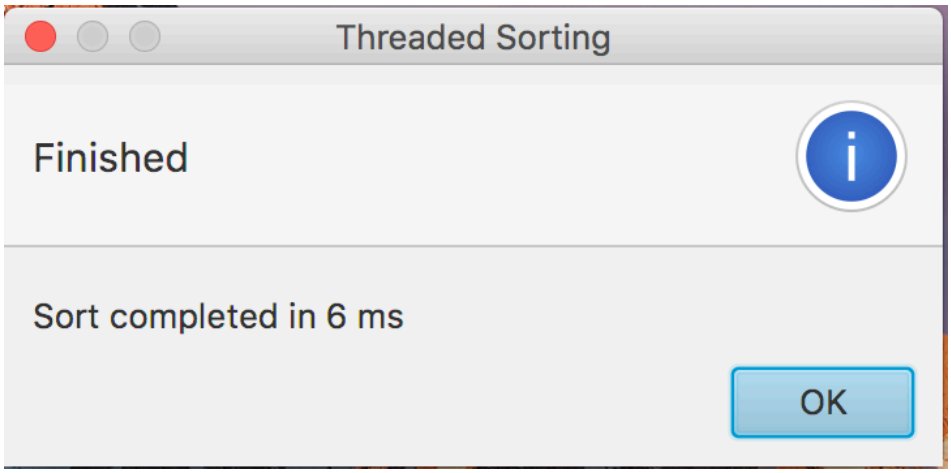
Example



The output to the console is this:

```
[0, 2, 2, 2, 3, 5, 5, 6, 7, 8, 9, 9, 13, 13, 15, 16, 17, 17, 19, 21,
23, 24, 25, 26, 27, 27, 32, 32, 32, 32, 33, 34, 35, 35, 36, 36, 37, 38,
38, 38, 39, 39, 41, 41, 43, 44, 45, 45, 46, 48, 48, 49, 50, 50, 53, 53,
54, 54, 54, 55, 55, 55, 60, 61, 62, 62, 63, 63, 63, 63, 65, 66, 68, 69,
69, 69, 72, 74, 74, 76, 76, 76, 78, 79, 80, 80, 81, 82, 82, 85, 85, 86,
87, 88, 89, 92, 95, 98, 98, 99]
```

And this alert dialog is displayed:



Your program will be graded according to this rubric (each item is worth one point):

- The program displays a user interface with the requested layout
- Pressing the Go button validates the user input and displays an alert dialog if necessary
- The program correctly implements selection, bubble, insertion, and quick sort
- The program correctly implements the merge method
- The program spawns threads to sort chunks of the array with the requested size and using the requested algorithm
- The program spawns threads to merge the sorted chunks
- The program displays the number of milliseconds the sorting operation took via an alert dialog