Programming Assignment 1 (Stack, Queue, Selection Sort, and Insertion Sort)

Department of Computer Science, University of Wisconsin – Whitewater
Data Structure (CS 223)

1 Stack and Queue

- Complete the *push*, *pop*, and *size* functions in the `Stack.java/Stack.h` files.
- Complete the *enqueue*, *dequeue*, and *size* functions in `Queue.java/Queue.h` files.

2 Queue Using Two Stacks

To showcase the fact that the same data structure can be implemented in multiple ways, we will use a queue and implement it using two stacks (as opposed to using an array).

Here's the main idea. *A stack reverses a sequence of items. A queue maintains a sequence of items. So, we use two stacks. One to reverse the sequence, and the other to reverse the reverse.* For more details, watch the `QueueUsingTwoStacks` video posted in the assignment folder.

Is that good? No, in fact terrible! It raises the complexity from $O(1)$ to $O(n)$, where n is the size of the queue.

So, why teach it? To make you understand the following – *just because it works does not mean it is correct or a good implementation.*

In any case, here's how to do it:

- Implement the *enqueue* function in the `QueueUsingStack.java/QueueUsingStack.h` file using two stacks as described below:
  - Create a stack "tempStack", which has the same maximum capacity as the mainStack.
  - Pop all numbers from mainStack and push them onto tempStack.
  - Push the new number onto mainStack.
  - Pop all numbers from tempStack and push them onto mainStack.
- Implement the *dequeue* function in the `QueueUsingStack.java/QueueUsingStack.h file`. This is nothing but a pop on the mainStack.
- Implement the *size* function in the `QueueUsingStack.java/QueueUsingStack.h file`. This is nothing but the size of mainStack.

3 Selection Sort and Insertion Sort

Complete the *selectionSort* and *insertionSort* functions of the `Sorting.cpp/Sorting.java` file.

4 Correctness

Use the `TestCorrectness.cpp`/ `TestCorrectness.java` files to test your code. The expected output is provided in `ExpectedOutput` file. You can use `www.diffchecker.com` to tally the output.

5 Time Test Output: Not Required for Grading

Use the TestTime.java/TestTime.cpp file to compare the two queue implementations. Note that enqueue function when implemented using an array has $O(1)$ complexity, and when using two arrays has $O(n)$ complexity, where n is the current size of the queue. So, the two stack implementation should take more time, which is substantiated by the experiment (time output).

```
Time taken for 20000 enqueue/dequeue operations, when using a stack implementation: 221
Time taken for 40000 enqueue/dequeue operations, when using a stack implementation: 690
Time taken for 60000 enqueue/dequeue operations, when using a stack implementation: 1160
Time taken for 80000 enqueue/dequeue operations, when using a stack implementation: 1845
Time taken for 100000 enqueue/dequeue operations, when using a stack implementation: 2881

Time taken for 2000000 enqueue/dequeue operations, when using an array implementation: 11
Time taken for 4000000 enqueue/dequeue operations, when using an array implementation: 20
Time taken for 6000000 enqueue/dequeue operations, when using an array implementation: 12
Time taken for 8000000 enqueue/dequeue operations, when using an array implementation: 13
Time taken for 10000000 enqueue/dequeue operations, when using an array implementation: 16
```