Hacettepe University
Computer Science and Engineering Department

Name and Surname : Murat Celik
Identity Number : 21827263
Course : BBM104
Experiment : Assignment 4
Subject : Stack and Queue Operations
Data Due : 10.06.2020 (23:59)

Problem :

Learn with data structures such as queue, stacks, etc. Learn this data structures's methods. Take an input file (command.txt) as an argument from the command line. According to this input file, make ***some operations*** and print the output.txt at the end of the input file operations.

Some Operations :

1) Remove Greater Number

Remove element which greater than input number in your data structure.

Stack :	Queue :
1) Create a new Stack 2) Remove a number in my stack 3) If the number in removed my stack less or equal than input number add in new Stack 4) Go to 2 until my stack is empty 5) Remove a number in new Stack and add the number in my Stack	1) Assign size of Queue to number of loop 2) Remove a number from front of Queue 3) If the number less or equal than input number ,add the number rear of Queue

2) Calculate Distance of Elements

Find the sum of the distances of all elements to other all elements in the current data structure

Stack :	Queue :
1. Create a new clone of my stack(oldStack). 2. Create a new stack(newStack) 3. If newStack is not empty, transfers newStack elements to oldStack. 4. Assign the number in top of oldStack to a first number 5. Add the first number to top of newstack 6. Assign the number in top of oldStack to a second number 7. Add the second number to top of newstack 8. Calculate distance of two number and assign to another number(sum) 9. Go to 7 until second loop is done. 10. Go to 3 until first loop is done.	1) Create a new clone of my queue. 2) Assign the number in front of Queue to a first number 3) Assign the number in front of Queue to a second number 4) Calculate distance of two number and assign to another number(sum) 5) Add the second number in rear of queue. 6) Go to 3 until second loop is done. 7) Go to 2 until first loop is done.

3. Add or Remove Elements

A value is read from the input file. If the number is negative, remove elements as the number of times. If the number is positive, add new random elements as the number of times

Stack:	Queue:
If the number is negative, use pop methods as the number of times . If the number is positive, create a new random number and use push methods as the number of times.	If the number is negative, use poll methods as the number of times . If the number is positive, create a new random number and use add methods as the number of times.
Last in first out. (top of Stack)	**First in (rear of Queue) first out. (front of Queue)**

4. Reverse Elements

Reverses the first X which according to given integer number in "command.txt" elements of stack or queue.

Stack:	Queue:
1. Get a input number(inNum) 2. Create a new Queue. 3. Remove numbers in my stack as the inNum of times and add these numbers in queue. 4. Remove numbers until queue is empty and add these numbers in my stack.	1. Get a input number(inNum) 2. Create a new Queue and new Stack. 3. Remove numbers in my queue as the inNum of times and add these numbers in stack. 4. Remove numbers until my queue is empty and add these numbers in new queue. 5. Transfer elements of stack then new queue to my queue.

5. Sorting Elements

All elements in the stack and queue are sorted

Stack:	Queue:
1. Create a new stack 2. Assign number top pf my stack to a number(lastNum) . 3. Start a loop, go on if my stack is not empty .(4-5-6) 4. Start a loop, go on if new stack is not empty and top of new stack greater than lastNum.(just 5) 5. Add top of new Stack in top of my stack. 6. Add top of my stack in top of new Stack. 7. Transfer elements new stack to my stack.	1. If size of my queue is even number, according to compare two number in front of queue and add these two numbers. This process should be with the aid of two loops. 2. If size of my queue is odd number, Do process similar to up. (minor differences)

6. Distinct Elements

Finds how many distinct elements there are in stack and queue.

Stack:	Queue:
1. Copy elements of my Stack to new Stack. 2. Sorted new Stack. 3. If two top of new stack is not equal, increase the counter 1.	1. Copy elements of my Queue to new Queue. 2. Sorted new Queue. 3. If two front of new queue is not equal ,increase the counter 1.

Analysis of removeGreater method in Queue class:

Time Complexity , Efficiency:

There are 2 for loops(Linear Complexity). There are many Constant Complexity.

Best case => 4n^2 + 18n + 15

Worst case => 4n^2 +22n+17

For very large numbers => 4n^2

If I used arraylist instead of array when writing queue methods, 1 for loop would be less and more efficient.