Programming Assignment 4

Submission Date : 22.05.2020
Due Date : 10.06.2020 (23:59)
Subject : Stack and Queue Operations

1 Problem

In this assignment, you will introduce with data structures such as queue, stacks, etc. You will take an input file (command.txt) as an argument from the command line. According to this input file, you will make some operations and print the output.txt at the end of the input file operations.

You will be given 3 input files namely command.txt, stack.txt (contains a maximum of 100 elements) and queue.txt (contains a maximum of 100 elements). You will update the data in the stack file and the queue file according to the each commands in the command.txt file. If the commands in the text file start with "S", that command is performed for the stack, but if it starts with "Q", it is performed for the queue. After each of the operations performed, the results also are printed into the stackOut.txt file for the stack and the queueOut.txt file for the queue.

1. Remove Greater Number

In this operation, a value is read from the input file then remove all numbers from the queue or stack which are greater than this number. After this operation, the final version of the stack or queue is printed on the their output files.

- Command format in the Command.txt file for this operation:

S[space] removeGreater [space] k (for stack)

Q[space] removeGreater [space] k (for queue)

Example:

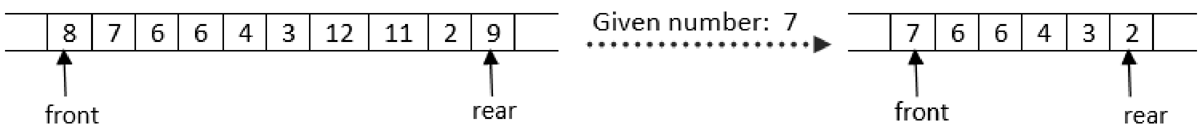


Figure 1: Content of structure after operation

You must use ONLY queue(s), don't use other data structures such as pure array structure, string, stack etc.

2.Calculate Distance of Elements

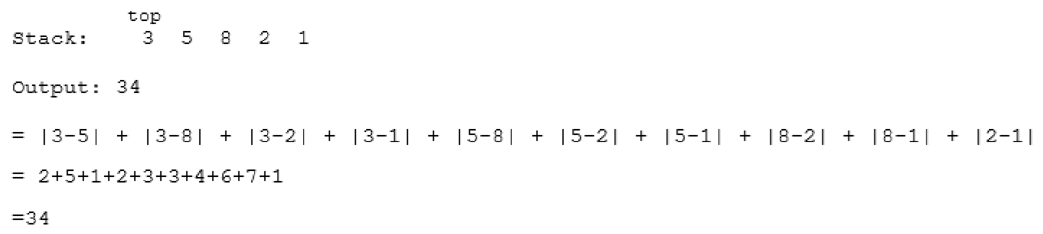
In this operation, finds the sum of the distances of all elements to other all elements in the current queue or stack file. Then, the results are then printed in the specified format in the queueOut.txt file for stack in stackOut.txt.

- Command format in the Command.txt file for this operation:

S[space] calculateDistance (for stack)

Q[space] calculateDistance (for queue)

Example:



```
            top
Stack:       3   5   8   2   1

Output: 34

= |3-5| + |3-8| + |3-2| + |3-1| + |5-8| + |5-2| + |5-1| + |8-2| + |8-1| + |2-1|

= 2+5+1+2+3+3+4+6+7+1

=34
```

Figure 2: Content of the structure after operation

3. Add or Remove Elements

In this operation, a negative or positive 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 (between 0-50).

- Command format in the Command.txt file for this operation:

S[space]addOrRemove[space]k(for stack)

Q[space]addOrRemove[space]k(for queue)

4. Reverse Elements

In this operation, according to integer k that is given in command.txt, reverses the first k elements of stack or queue. After this operation, the final version of the stack or queue is printed on the their output files.

- Command format in the Command.txt file fot this operation:

S[space]reverse[space]k(for stack)

Q[space]reverse[space]k(for queue)

5. Sorting Elements

In this operation, all elements in the stack and queue files are sorted. You can use any sorting operation you want.

- Command format in the Command.txt file for this operation:

S[space]sortElements(for stack)

Q[space]sortElements(for queue)

6. Distinct Elements

In this operation, you are expected to finds how many distinct elements there are in stack and queue files.

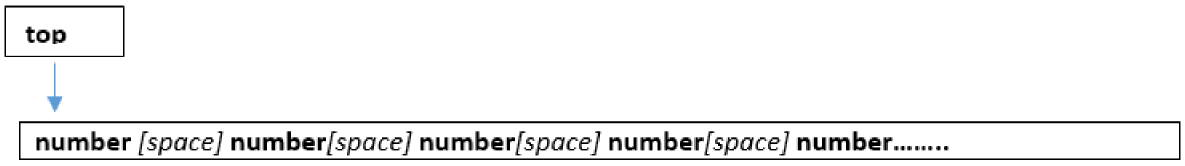
- Command format in the Command.txt file for this operation:

S[space]distinctElements(for stack)

Q[space]distinctElements(for queue)

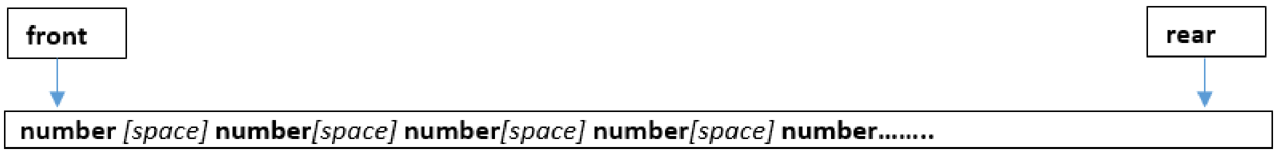
2 Input Files

Below, you can find an example of input and output file format.



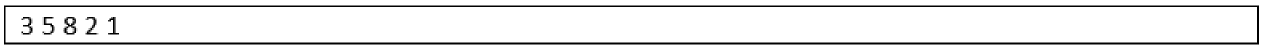
number *[space]* **number***[space]* **number***[space]* **number***[space]* **number……..**

Figure 3: format of stack.txt



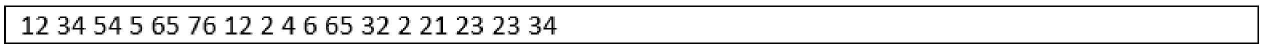
number *[space]* **number***[space]* **number***[space]* **number***[space]* **number……..**

Figure 4: format of queue.txt



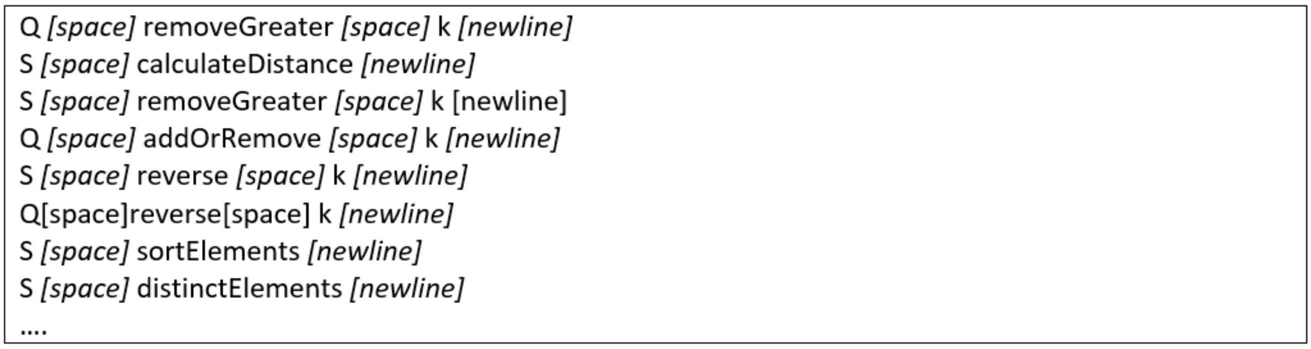
```
3 5 8 2 1
```

Figure 5: example of stack.txt



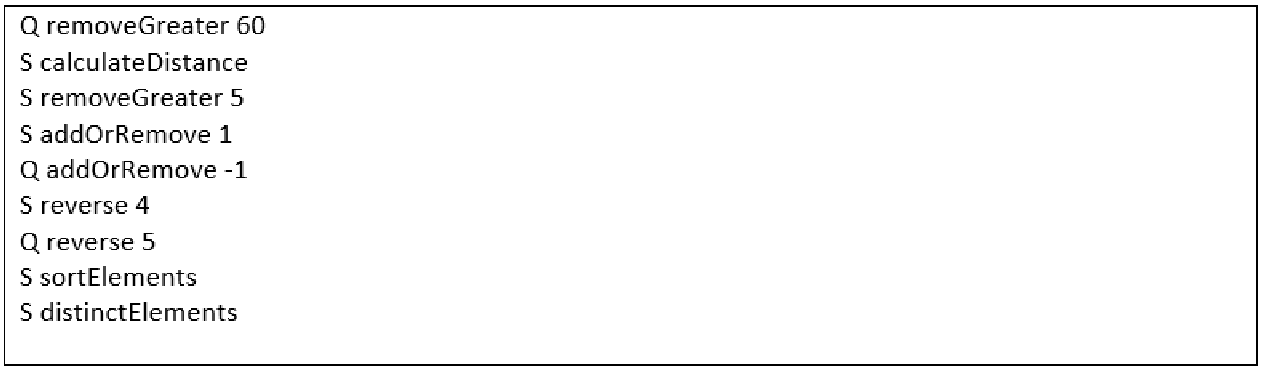
```
12 34 54 5 65 76 12 2 4 6 65 32 2 21 23 23 34
```

Figure 6: example of queue.txt



```
Q [space] removeGreater [space] k [newline]
S [space] calculateDistance [newline]
S [space] removeGreater [space] k [newline]
Q [space] addOrRemove [space] k [newline]
S [space] reverse [space] k [newline]
Q[space]reverse[space] k [newline]
S [space] sortElements [newline]
S [space] distinctElements [newline]
....
```

Figure 7: format of command.txt

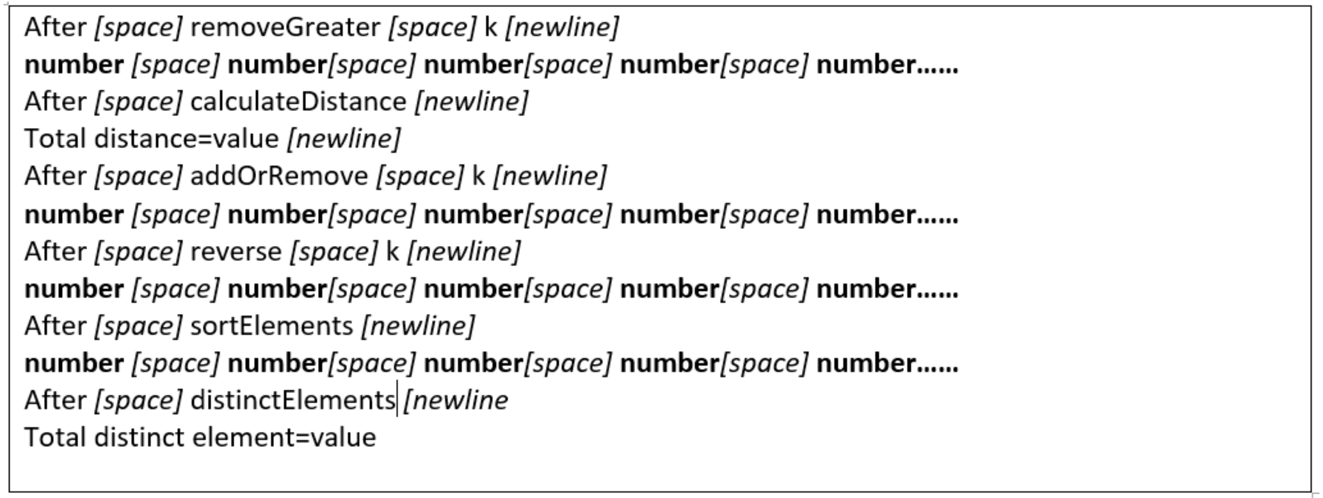


```
Q removeGreater 60
S calculateDistance
S removeGreater 5
S addOrRemove 1
Q addOrRemove -1
S reverse 4
Q reverse 5
S sortElements
S distinctElements
```

Figure 8: example of command.txt

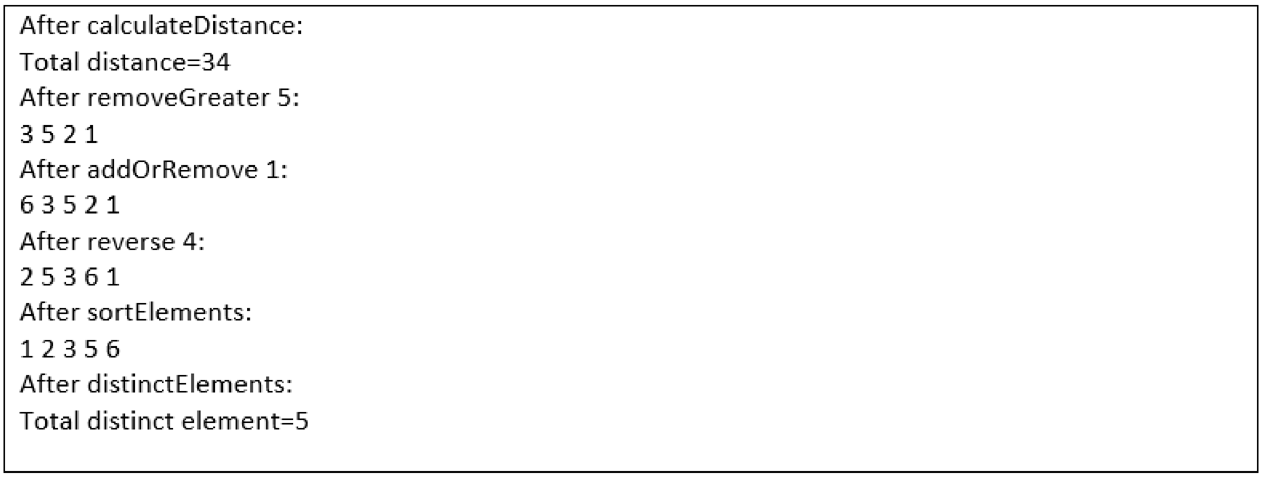
3 Output Files

Output files will be produced separately for the stack and queue as stackOut.txt and queue-Out.txt.



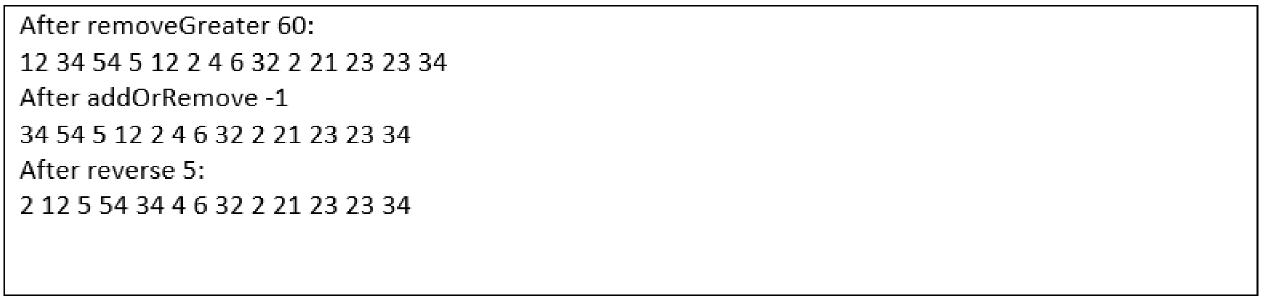
After *[space]* removeGreater *[space]* k *[newline]*
number *[space]* **number***[space]* **number***[space]* **number***[space]* **number……**
After *[space]* calculateDistance *[newline]*
Total distance=value *[newline]*
After *[space]* addOrRemove *[space]* k *[newline]*
number *[space]* **number***[space]* **number***[space]* **number***[space]* **number……**
After *[space]* reverse *[space]* k *[newline]*
number *[space]* **number***[space]* **number***[space]* **number***[space]* **number……**
After *[space]* sortElements *[newline]*
number *[space]* **number***[space]* **number***[space]* **number***[space]* **number……**
After *[space]* distinctElements *[newline*
Total distinct element=value

Figure 9: format of stackOut.txt and queueOut.txt



```
After calculateDistance:
Total distance=34
After removeGreater 5:
3 5 2 1
After addOrRemove 1:
6 3 5 2 1
After reverse 4:
2 5 3 6 1
After sortElements:
1 2 3 5 6
After distinctElements:
Total distinct element=5
```

Figure 10: example of stackOut.txt



```
After removeGreater 60:
12 34 54 5 12 2 4 6 32 2 21 23 23 34
After addOrRemove -1
34 54 5 12 2 4 6 32 2 21 23 23 34
After reverse 5:
2 12 5 54 34 4 6 32 2 21 23 23 34
```

Figure 11: example of queueOut.txt

4 Execution and Test

- Upload your java files to your server account (dev.cs.hacettepe.edu.tr)
- Compile your code (javac *.java)
- Run your program (java Main command.txt)
- Control your output.

5 Submit Format

- File hierarchy must be zipped before submitted (Not .rar, only .zip files are supported by the system)

- ⟨*studentid*⟩*.zip*
– *src*(*Main.java*, ∗*.java*)

6 Grading Policy

Task	**Point**
Submit	1
Compiled	10
Clean code	15
Coding standard	5
Output	59
Report	10
Total	100

7 Notes

- **You MUST implement your own queue and stack classes, don't use other data structures such as array, string, arraylist, linkedlist etc in your imple-**

mentation. If you use other data structures, you will not get any point (your grade will be 0).

- The assignment must be original, individual work. Downloaded or modified source codes will be considered as cheating. Also the students who share their works will be punished in the same way.
- We will be using the Measure of Software Similarity (MOSS) to identify cases of possible plagiarism. Our department takes the act of plagiarism very seriously. Those caught plagiarizing (both originators and copiers) will be sanctioned.
- You can ask your questions through course's piazza group and you are supposed to be aware of everything discussed in the piazza group. General discussion of the problem is allowed, but DO NOT SHARE answers, algorithms, source codes and reports .
- With this assignment which covers the subjects of data structures like stacks, queue, etc; you are expected to gain practice on the basics of these structure. Therefore, you will not be graded if any paradigm other than these is developed!
- **In your REPORT,** it is important giving explanation about problem definition (limited to max. 3 sentences) and steps of algorithms that you followed (limited to max. 5 sentences). Do not give so much unnecessary details in your report (totally max 2 pages written with Times New Roman 12). Also, it is expected you to make an analysis of one of commands given in commands.txt.
- Don't forget to write comments of your codes when necessary.
- The names of classes', attributes' and methods' should obey to Java naming convention.
- Save all work until the assignment is graded.
- Do not miss the deadline. Submission will be end at 10/06/2020 23:59, the system will be open 23:59:59. The problem about submission after 23:59 will not be considered.
- There will be again 3 days extensions (each day degraded by 10 points) in this project.