## HACETTEPE UNIVERSITY

DEPARTMENT OF COMPUTER ENGINEERING BBM104



Name :

Surname :

Number :

Subject : Stack and Queue Operations

Programming Language : Java

### Problem Definition

Introduce with data structures such as queue, stacks

### Problem Solving

Firstly, The reader object is created to read the files. After creating the reader object, the reader object reads the file specified in the user parameter. Queue.txt is read using the read\_queue () function in the reader object. Numbers in the row are stored in the queue\_structure variable with Queue methods. Stack.txt is read using the read\_stack () function in the reader object. The numbers in the row are stored in the stack\_structure variable with Stack methods. Finally, using the process () function, the lines of the file in the parameter are processed. each row is separated by space. Decides whether the array that is formed after split is stack or queue according to the first element. Necessary operations are carried out according to the second and third element. If the second element is removeGreater, the larger elements are deleted from the third element. If the second element is calculateDistance, calculations are made with the for loop. If the second element, addOrRemove, is processed according to the third element. If the third element is positive, a random number between 0-50 is added (in stack or queue). If the third element is negative, the number is deleted as much as the third element (in stack or queue). If the second element is Reverse, the number is reversed as much as the third element. The second element, distinctElements, determines how many different numbers are using hashmap. If the second element is sortElement, the elements are sorted from small to large. Results are written to txt (QueueOut.txt and StackOut.txt).

1. Stack analysis

Push():Time complexity O(1).

Pop():Time complexity O(1).

Peek():Time complexity O(1).

isEmpty(): Time complexity O(1).

isFull():Time complexity O(1).

copy(): Time complexity O(N).

sort(): Time complexity O(N\*log(n)).