Luis Lopez
Assignment 2

Assignment 2 – Stacks and Queues

i.

- Create all Items in a loop with user data size as the loop controller
- Use a while loop to start filling up the queue with each induvial Item until full.
- Now that the itemQueue is full I can start the process of organizing the items into the two stacks to help me sort them into the finalOutputArray
- The main algorithm starts here:
    - I initially assume the first in queue has the max Key
    - Grab the next Item in queue and compare with MaxKey
      Here are 3 scenarios
- Item in queue is greater than MaxKey
    - Insert MaxKey item back into queue and replace remove item in front to MaxKey
- Item in queue is less than MaxKey
    - Leave MaxKey as it is, then remove item in front of queue to the back.
- Item in queue equals MaxKey
    - Push MaxKey into the TempStack and remove item in front of queue into MaxKey old place.
- Keep comparing MaxKey and item of front of queue until it reaches the Item with the max Keystamp.
- At this point, if they are any items in TempStack, start popping them and pushing them into MainStack.
- If they are no items in TempStack, push the MaxKey item into the MainStack.
- Then we start all over again and continue until the queue is empty.
- After all that, MainStack should have all the items sorted correctly.
- Then I simply make a for loop with size of the stack as the loop controller, starting from first element, start popping each item from MainStack into the finalOutputArray.

ii. Queue played the biggest role in my algorithm because it allowed me to cycle through it many times and compare the maximum with the front of the queue in order to find the max key. With three stacks, maybe I could have come up with a different algorithm to fulfill the requirements.

iii. I believe my algorithm is not stable. When MaxKey and first in queue equal each other, there's a chance that the TimeStamps get unordered when I keep cycling through the Queue.

iv. This experimental sorting algorithm does sort in place. By only using two stacks and one queue, it allowed me to use a lot less memory than using traditional arrays.

v. I believe the running time of this algorithm is O(1) because moving through Stacks and Queues to pop/push or remove or insert does not require going through its array at all.

vi. The biggest challenge of making this algorithm was coming up with it by being limited to two Stacks and a Queue. It was very hard to find the pattern that would make sense and that could also translate well into code. The second challenge was the actual translation into code and making sure that all everything worked as it should, which is many cases it did not, and I couldn’t figure out why. But overall, this was a very challenging assignment and I have learned a lot from it.