Shaheryar Syed 30052162
Harnoor Khehra Tutorial T04

ASSIGNMENT 2 – Exercise 3 WRITEUP:

For starters, since we were not allowed to use java.util library, I assumed it was not possible to use something like a HashSet which would allow me to use the .contains method. As such, I had to manually check each item in the stack and compare it with items inside uniqueSet. Therefore, the time complexity would be different for this algorithm under the assumption that we are not allowed to use the hashset.

When considering the following algorithm:

```
        int[] uniqueSet = new int[12];
        int uniqueCount = 0;
        QueueImplementation uniqueQueue = new QueueImplementation(9);
        for(int i = 0; i < stacker.arraySize; i++) {
            int value = stacker.returnSingle(i);
            boolean isUnique = true;

            /* Check to see if the value is unique */
            for (int j = 0; j < uniqueCount; j++) {
                if(uniqueSet[j] == value) {
                    isUnique = false;
                    break;
                }
            }

            /* If isUnique is true, add that value to the queue and uniqueSet */
            if(isUnique) {
                if(uniqueCount < uniqueSet.length) {
                    uniqueSet[uniqueCount++] = value;
                    uniqueQueue.enqueue(value);
                }
            }

        }
```

We can see that the first 3 lines are O(1) as they are in linear time.

For the first few lines in the for loop, we are only iterating over the stack and grabbing it's value, as such, the for loop makes the first section's Big O notation O(n), as the value line, and the Boolean line is in linear time, namely O(1). However, the for loop executes n times so the big O notation is O(n).

If we look at the second part where we are checking if the value is unique, we see that there is another for loop. We know from lectures that for loops have a complexity O(n). As such, we now have a for loop with a complexity of O(n) followed by another for loop with a time complexity of O(n). If we move down to the if(isUnique) statement, its clear that this whole part follows a linear

complexity that is a time complexity of O(1) as for loops or if statements don't affect the time complexity n times. As such, we have O(n*n) which results in a time complexity of O(n^2) for this algorithm.