Matthew Jackson Problem 2 11/2/2018

1. The remove() function I used does not access an array of any size. Since remove() directly calls a value, it runs in O(1) time. In my remove() function below, the integer “item” is being directly accessed and returned. Following the same logic, my getValue(int i) function also runs in O(1) time because it directly calls on items in the queue.

Remove: public int remove(){

int item = first.item;

first = first.next;

sizeOfQueue--;

return item;

}

getValue: public int getValue(int i) {

if (i == 0){

return first.item;

} else if (i == 1) {

return first.next.item;

} else if (i == 2) {

return first.next.next.item;

} else if (i == 3) {

return first.next.next.next.item;

} else if (i == 4) {

return first.next.next.next.next.item;

} else if (i == 5) {

return first.next.next.next.next.next.item;

}

return -1;

}

2. My add() function has a time complexity of O(N) because of the time it takes to input a value into the queue and update the next to last value in the queue. As seen in the code below, last.next needs to be updated to null after an item is inputted. If a value could be inserted into the queue without having to update any existing values, it would run in O(1) time. Since this is not the case, the time complexity is O(N).

Add: public void add(int item){

Node oldlast = last;

last = new Node();

last.item = item;

last.next = null;

if (isEmpty()) {

first = last;

} else {

oldlast.next = last;

}

sizeOfQueue++;

}