1. The remove function removes the node at the head of the list. The code is here:

**public** **int** remove(){

**int** temp = head.val;

System.***out***.println("removing " + temp);

head = head.next;

numRemoved++;

**return** temp;

}

Because the first in node is the head of the list, is is the only one you need to access so it can always be accessed in constant time. The only thing to do is make the next node the new head.

The getValue function is here:

**public** **int** getValue(**int** i){

**if** (i > numValues + numRemoved -1){

**return** -1;

}

**return** storedValues[numRemoved + i];

}

There is an array that contains every single element that has been entered. As elements are removed from the structure they stay in the array and are just skipped over. So the index of the element in the array is the same as the index of whatever you’re looking for plus the number of elements that have been removed.

2.

Everything the add function does is done in linear time or constant time, and the sums of all of these operations will therefore be in at worst linear time. First it checks if there is a need to increase the size of the array that stored all of the elements, and if there is it makes a new, bigger array and copies the data over which happens in linear time. Then it checks if the list is empty, and if it is then it makes the head the new element and this is constant time. If the queue is not empty, it increments through the queue until it hits the end, which happnes in linear time, then adds the element by making anew node, which is a constant time operation.

Here is the code:

**public** **void** add(**int** a){

**if**(numValues + numRemoved == storedValues.length){

**int**[] newStoredValues = **new** **int**[storedValues.length\*2];

**for**(**int** i = 0; i < numValues; i++){

newStoredValues[i] = storedValues[i];

}

}

**if** (numValues == 0){

head.val = a;

}

**else**{

Node x = head;

**while**(x.next != **null**){

x = x.next;

}

Node newNode = **new** Node();

newNode.val = a;

x.next = newNode;

}

storedValues[numValues+numRemoved] = a;

numValues++;

}