Problem 3

1. This solution achieves O(logN + N) time since it implements the binary search algorithm.

The binary search algorithm splits the data structure into two over and over until the value is found or all values have been checked:

**while**(start <= end)

{

**int** mid = (start+end)/2;

when compared to the value being searched for if the value in the data structure at mid is greater, then only the lower half of the data structure continues to be checked, if it is smaller only the greater half continues to be checked:

**if**(d.getValue(mid) > val)

{

end = mid -1; ←- this ensures only the lower half is checked

}

**if**(d.getValue(mid) < val)

{

start = mid + 1; ←- this ensures only the higher half is checked

}

This gives logN time.

O(N) time is needed to find what the end value should be:

**int** end = 0;

**while**(d.getValue(end) != -1)

{

end++;

}

This will run N times for N elements.