Activity #1: Selection. Try to hand simulate select(list, k) with the given inputs, and a random pivot (in this case we'll use the first element):

listy = [2,8,1,5,4,1,3,9],
select(listy, 4):

Activity #2: What is the worst case runtime of Hoare’s Selection?
What case gives us this runtime?

Activity #3: Provide the worst-case runtime for each line in Median of Medians Select:

(Hint: Remember that because our median is guaranteed to be between the 25th and 75th percentiles, we can be sure that in the worst case it will divide the input list into lists of size 3n/10 and 7n/10 on each call)

```
momSelect(list, k)
    // Base case omitted
    miniLists = divide list into n/5 lists of 5 ---- #1
    medians = []
    for miniList in miniLists:
        sort5(miniList)                        #2
        medians.append(miniList[2])           #3
    pivot = momSelect(medians, medians.size/2) #4
    L = []   E = []   G = []
    for x in list:
        if x < pivot: L.append(x)
        if x == pivot: E.append(x)
        if x > pivot: R.append(x)
    if k <= L.size:
        return momSelect(L, k)               #7
    else if k <= (L.size + E.size)
        return pivot
    else
        return momSelect(G, k - (L.size + E.size))  #8
```

Now that we know the line by line runtime breakdown of momSelect, can we determine a general recurrence relation:

<table>
<thead>
<tr>
<th>Runtimes</th>
<th>1: O(____)</th>
<th>2: O(____) (loop total)</th>
<th>3: O(____)</th>
<th>3: O(____)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4: O(____)</td>
<td>5: O(____)</td>
<td>6: O(____) (loop total)</td>
<td>7: O(____)</td>
<td>8: O(____)</td>
</tr>
</tbody>
</table>