1. Show the full contents of the internal arrays for s1 and s2 after each of the following statements are executed. The first few statements are completed for you.

<table>
<thead>
<tr>
<th>s1's array</th>
<th>s2's array</th>
</tr>
</thead>
<tbody>
<tr>
<td>nullptr</td>
<td>nullptr</td>
</tr>
<tr>
<td>{b}</td>
<td>nullptr</td>
</tr>
<tr>
<td>{b, a}</td>
<td>nullptr</td>
</tr>
<tr>
<td>{b, c, a, -}</td>
<td>nullptr</td>
</tr>
</tbody>
</table>

ArraySeq<char> s1, s2;
s1.insert('b', 0);
s1.insert('a', 1);
s1.insert('c', 1);
s2 = s1;
s2.insert('d', 0);
s1.insert('f', 2);
s1.insert('e', 4);
s1.erase(1);
s2 = std::move(s1);

2. Use the array-based version of the merge sort algorithm from class to sort the array \{6, 3, 5, 7, 1, 4, 2\}. Show the array after each “split” and “merge” phase of merge sort. The first “split” step is shown below.

original: \{6, 3, 5, 7, 4, 1, 2\}
split: \{6, 3, 5, 7\} \{4, 1, 2\}
3. Use the array-based version of the quicksort algorithm from class to sort the array \{4, 3, 5, 7, 1, 6, 2\}. Assume we also pick the first element of the (sub) array as the pivot value. Show the array after each stage. (Note you need to follow the pseudocode for this question.)

4. Assume that instead of creating an array that is twice the size of the original, the resize() function creates an array that is three-times the original size. Show that the insert-at-end function is still amortized $O(1)$. 