Question 1: What is this general style of algorithm called?

*Divide and Conquer (a kind of recursive algorithm)*

Question 2: What happens if `Freq` is called on collection `S` where two different elements both have the maximum frequency?

*It might seem at first like the larger of the two is always returned, because `f_high` “wins” if there is a tie, in Step 4. However, if the larger element is chosen as `e`, then `Freq` will return the smaller element.*

Question 3: Suggest a method to perform Steps 2 and 3 of `Freq` with one pass through `S` (not counting the recursive calls). What data structure for `S` would support that method?

*We don’t need to go through `S` twice to get `S_low` and `S_high`. We can make one pass, placing each element in the appropriate subcollection based on comparison with `e`. Linked lists would be good for this approach. We can also compute `f` and the sizes of `S_low` and `S_high` during this pass.*

*(It’s also possible to do the partitioning into `S_low` and `S_high` in place in an array, as is done in some implementations of Quicksort.)*

Question 4: Suggest an improvement to Step 3 that can avoid one or both recursive calls in some cases.

*If `f > |S_low|`, we can just use `(null, 0)` for `(e_low, f_low)` instead of calling `Freq(S_low)`, because `S_low` can’t contain an element that is more frequent than `e`. Similarly for `S_high`.*

Question 5: Suppose we are fortunate and the choice of `e` always splits `S` into subsets of approximately equal size (that is `|S_low| = |S_high|`). What is the maximum depth of recursion `Freq` will have on an input of size `n`?

*In the case described, at each level of recursion, we will have calls on collections that are roughly half the size of the input. The number of times we can divide `n` in half is roughly `log_2 n` (also known as `lg n`).*
The following routine finds the most frequent element in a collection $S$ of integers, plus the frequency (number of occurrences) of that element (without having to sort $S$). Note that $S$ can have duplicates.

$|S|$ means the number of elements in $S$. For example $\{|3, 9, 8, 3|\} = 4$.

Freq($S$)
0. If $|S| = 0$, return (null, 0)
1. Pick some element $e \in S$.
2. Split $S$ into two subsets:
   \[
   S_{low} \leftarrow \{d \in S \mid d < e\}
   \]
   \[
   S_{high} \leftarrow \{d \in S \mid d > e\}
   \]
3. Set $f = |S| - |S_{low}| - |S_{high}|$
   Set $(e_{low}, f_{low}) \leftarrow \text{Freq}(S_{low})$
   Set $(e_{high}, f_{high}) \leftarrow \text{Freq}(S_{high})$
4. If $f > f_{low}$ and $f > f_{high}$, return $(e, f)$
   else if $f_{low} > f_{high}$, return $(e_{low}, f_{low})$
   else return $(e_{high}, f_{high})$

Show a sequence of recursive calls for
Freq({4, 2, 1, 6, 6, 1, 2, 2, 8, 4})