1a. Only the most upfront value can be accessed in a queue.
1b. Only the last value added can be accessed in a stack.
2. A list backed by links would be better if a lot of adding a removing were involved in the list because adding and removing at a particular index has a constant asymptotic complexity while this complexity is linear when using arrays since the array has to be recreated every time the method is called.

3a. The complexity is linear because the array has to be recreated every single time the append method is called.
3b. The complexity is linear because the array has to be recreated every single time the append method is called.
3c. The complexity is constant because arrays are able to fetch a value at any index.

4a. Appending is constant in a link list because you can just go to the tail and add the new node as the tail.
4b. The complexity is linear because you would still have to go through the list to get to the index of the last fetched value and remove it.
4c. The complexity is linear because as the list index grows larger the longer it'll take to get through each node to get to that index.

5a. The complexity is linear because you would need to go through every value in the list and check if the value in the list is equal to the value you are looking for.
5b. The complexity is also linear because you need to do the exact kind of thing as the array list.
5c. The upper bound for a binary search tree is linear because the worst case scenario is that the search goes through every single item going a long linear line. The lower bound is logarithmic because every time you go down the tree the amount is split by half the amount of nodes.
5d. If the tree is complete the type bound becomes logarithmic because the depth of each branch is the same making going down each branch halving the amount of nodes available to search.

6. A binary search tree can be better since the upper bound is linear and the lower bound is logarithmic, making it in most cases faster.