Recursion

1 In your own words, what is recursion?

Recursion is when in order to obtain the result we need to call the function inside the function to solve a smaller problem that looks exactly the same as the one we are trying to solve. For example if we are trying to add two numbers (num1 and num2) and we know what num1 is but in order to find out num2 we need to add two numbers. :)

2 Why is it necessary to have a base case?

It is necessary to know when the function needs to stop calling itself. Without a base case it will go into an an infinite loop.

Graphs

1 What is a graph?

graphs are like trees but they have cycles.

2 How is a graph different from a tree?

They have cycles. So I could circle back to a node using by moving from node to node. Also graphs don't have hierarchy

3 Give an example of something that would be good to model with a graph.

Relationship between friend in Facebook. If someone accepts my friend request they are my friends and I automatically become their friends as well.

Performance of Different Data Structures

Fill in the missing spots in the chart with the correct runtimes. Do this by reasoning through how the data structures work, **NOT** by looking up the solution. **Add-R** means add to the right/end/top and **Add-L** means add to the left/beginning/bottom. There are X's in the spots where that operation doesn't make sense for that data structure (for instance, you can't index a stack, or pop from the end of a queue). We've provided the first few answers for you.

Linked List	*O(n)*	*O(n)*	*O(1)*	*O(1)*	*O(1)*	*On*
Doubly-Linked List	*O(n)*	*O(n)*	*O(1)*	*O(1)*	*O(1)*	*O(1)*
Queue (as Array)	*X*	*X*	*O(n)*	*X*	*O(1)*	*X*
Queue (as LL or DLL)	*X*	*X*	*O(1)*	*X*	*O(1)*	*X*
Stack (as Array, LL, or DLL)	*X*	*X*	*O(1)*	*X*	*X*	*O(1)*
Deque (as DLL)	*X*	*X*	*O(1)*	*O(1)*	*O(1)*	*O(1)*

Table 1-1

Data Structure	***Get***	***Add***	***Delete***	***Iterate***	***Memory***	
Dictionary (Hash Map)	*O(1)*	*O(1)*	*O(1)*	*O(n)*	*medium*	
Set (Hash Map)	*O(1)*	*O(1)*	*O(1)*	*not sure.*	*medium*	
Binary Search Tree	*O(log n)*	*O(log n)*	*O(log n)*	*O(log n)*	*medium*	
Tree	*O(n)*	*O(1)*	*O(1)*	*O(n)*	*small*	Not counting with the search that needs to be done to add or delete a node just the add to delete action

Sorting

1 Describe in words how the Bubble Sort algorithm works.

It makes the “greatest” or “bigger” item bubble up to the last position.

2 Describe in words how the Merge Sort algorithm works.

we have two sorted lists and compare the first item in list1 to the first element in list2 and put them in the correct order in a results list and keeps doing it until the end of both lists. If one list is smaller than the other we append the rest of items in the longer list to the results lists (since they are already ordered we don’t need to worry about them.

3 Describe in words how the Quick Sort algorithm works.

We randomly pick a pivot and find the right place in the list for the pivot (item to the left is smaller and item to the right is greater). In order to achieve this we need two markers that will start at the begging of the list. **Marker1** will check all the items in the list and **Marker2** will mark the place where items on the left to this marker are smaller than the pivot and items on the right are greater than the pivot. **Marker1** will keep going item by item and if they will be compare to the item the pivot if it is lower it will move where Marker2 and so forth. Once Marker1 gets to the end of the list the pivot will be place where Marker one is.