Problem Set 8, Part I

Problem 1: Hash tables

1-1) linear

0	ant
1	flea
2	bat
3	cat
4	goat
5	dog
6	bird
7	bison

duck

1-2) quadratic

0	
1	
2	
3	cat
4	goat
5	bird
6	bison
7	dog

ant

1-3) double hashing

0	ant
1	bat
2	flea
3	cat
4	goat
5	bison
6	bird
7	dog

duck

1-4) probe sequence:
3,6,1,4,7,2

1-5) table after the insertion:

0	
1	leopard
2	
3	fly
4	toad
5	
6	cat
7	terrier

Problem 2: Comparing data structures
Both data structures would work (roughly speaking) equally well. Let's look at the points:

For the first bullet point, hash tables take the point since you can't make a regular balanced 2-3 tree to spell something out, and you'd need to get creative to make another function that turns the names of the products into numbers which also takes up more unnecessary processing time.

For the second bullet point, both methods take half a point since if we make the hash function revolve around the first n letters of the key, we can return a queue of values in the hash table associated with that key. Similarly, if we manage to make a balanced tree out of words, we can group words that start with the same prefix together which helps us find possible records that work with those letters like Google's or Apple's predictive text does.

For the third point, hash tables take the point again since looking up something in a hash table has an O(1) time complexity which is less than a balanced tree's O(logn).

For the fourth point, the balanced tree wins the point since its size can be increased indefinitely while a hash table has a fixed size and can overflow.

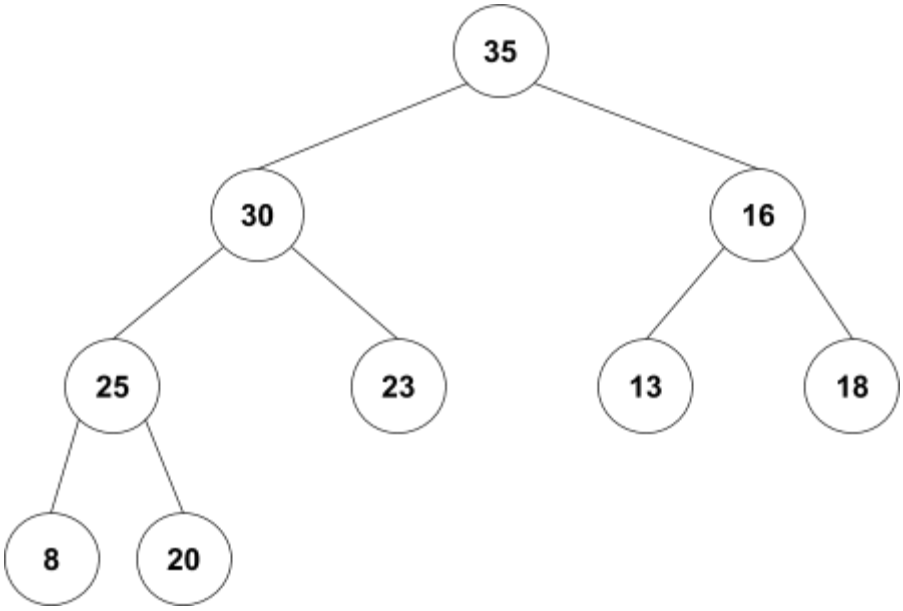
Overall, the hash table wins with its 2.5 points against the balanced tree's 1.5 points. However, since the last bullet point seems very important and can definitely not be replicated with a hash table, we can say that both of these methods are roughly equal in terms of the best choice.

Problem 3: Complete trees and arrays
3-1) They are 101, 102, and 24, and we got them using the following formulas where given node a[i], the following can be gotten:
left child=a[2i +1]
right child = a[2i +2] or leftchild+1
parent = a[(i-1)/2] where integer division is used

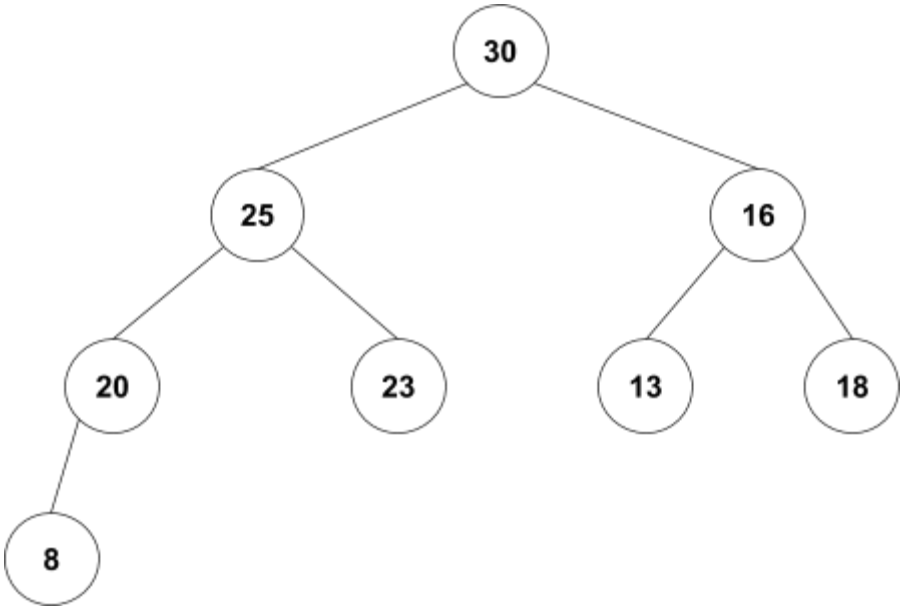
3-2) There's 7 layers. You can get the amount of nodes in a layer by the formula 2^(layer number) where the layer number for the root is 0. Hence, since 2^(0)+2^(1)...2^(7) equals 255 and 2^(0)+2^(1)...2^(6) equals 127, then tree must have 7 layers since there's not enough space for 200 nodes in 6.

3-3) It's a left child, since any node with an odd index is a left child. You can see that in the pattern where 0=root, 1=left, 2=right, 3=left, 4=right,....one induction later…… 199=left

Problem 4: Heaps
4-1)
after one removal



after a second removal
(copy your revised diagram from part 1 here, and edit it to show the result of the second removal)



4-2)