1) a) True. if \( f(n) = \Theta(g(n)) \) and \( g(n) = \Theta(f(n)) \)

b) True. if \( f(n) = O(h(n)) \) and \( h(n) = \Omega(f(n)) \)

c) False. \( O(\cdot) \) only indicates that the two are asymptotically equivalent; they may have different constants.

d) True. They are asymptotically equivalent.

e) True. The list goes through \( n \) times, each going another \( \frac{1}{2}n \) times.-motion.

False, slicing takes linear-time.

2) First, create a hash table to store unmatched elements. Next step through the two linked lists, in alternating sequences. When you have reached the end of a list, if the other still has elements (unless it only has one element left and is the second LL) return false. Otherwise, both LLs have the same size. Check now that the counter = 0 and return true if it does.
3) \( p = (1 - \frac{1}{m})^n \)

4) a) True:

b) The smallest element must reside in a leaf node, or \( i \geq \left\lceil \frac{\log n}{2} \right\rceil \), for slot \( i \) in a node.

c) i) Each insert operation runs in \( O(\log n) \) time, or the height of the heap tree, since it involves swapping up \( \log n \) times in the worst-case. With \( n \) insertions, the running time is \( n \log n \).

ii) In the worst case, each element takes \( \log n \) times swaps to insert, which occurs \( n \) times.

5) a) Splicing a list, \( O(n) \)

b) False, the keys must be known in advance.

c) \[ T(n) = \begin{cases} O(1), & \text{if } n = 1 \\ T(n-1) + T(n-1), & \text{else} \end{cases} \]
Substitution Method, guess $T(n) = O(2^n)$

$T(n) < 2^n$

Base Case: $O(1) < 2^1$ TRUE

 Recursive Case:

$T(n-1) + T(n-1) < 2^n$

RECURRENCE TREE

4) $T(n) = T(n-1) + O(1)$

$\sum_{i=1}^{n} = O(n)$

5) $T(n) = T(n-1) + \log T(n-1)$

$T(n) = \Theta(n \log n)$
b) False, worst-case is $O(n)$ if all keys hash to the same slot.

b) False, the probabilities of a slot being filled are no longer independent.

c) $m$ is not prime, therefore many keys will hash to the same slot if they are a factor of $m$. 
QUIZ #2:

1) False, comparison sorts require \( n \log n \) in the worst case which
\[ \frac{5}{8} < 6 \]

2) False. Heap sort is not a stable sort.

3) False. \( \frac{\text{finishes after}}{\text{ }} \)

4) False. Consider \( \frac{\text{ }}{\text{ }} \)

5) False, only with a stable sort.

6) True. \( \frac{\text{ } \text{False, non-negative edges}}{\text{ }} \)

7) True.

2) The size of the first \( \log n / 2 \) levels of the BFS tree was \( \ll \) than the second \( \log n / 2 \) levels
\[ \Delta > \Delta + \Delta \]

Hilroy
2) \( O(V + E) \) Dijkstra's + BFS

3] First search for the first degree zero node.

Pick a node \( s \), trace backwards along its edges until you.

Do a BFS backwards from any starting node \( s \), onto its incoming edges, then when you find a head-end, pop it and remove its edges.

If \( G \) has cycles this returns an infinite loop.

4) 1)

2) Check that:

Backtrack through all \( P \) to ensure that they never repeat.
3) Run Bellman Ford on each node, and return the maximum value found, then

Run Bellman Ford and return the longest shortest path, then repeat this for each node, returning the maximum of that

BF has running time of $O(VE)$
Therefore this algorithm is $O(V^2E)$ since it runs BF $V$ times $O(V^3)$ if $n = V + E$

4) Beginning at $s$, run a BFS which only considers vertices which are $\leq \delta(s, u) \leq \delta(s, t)$

*Begin at $t$!*