CSCI 4041, Spring 2019, Quiz 3 (30 minutes, 20 points)

Name:

x500:

Discussion Start Time (**circle one**): 3:35 4:40 5:45 6:50 7:55 other:

---

1. (1 points each) True/False - Circle one. Note that when asking about the properties of an algorithm, we specifically mean the version of that algorithm discussed in lecture.

   True False All comparison sorts have a Θ(n lg n) worst case runtime.

   True False If Insertion Sort is used as the algorithm within Radix Sort to sort by each digit, Radix Sort may fail: the output array may not be in sorted order.

   True False When using a hash table implemented with a doubly-linked-list that resolves collisions by chaining, the worst case runtime of deletion is Θ(1).

   True False The hash function $h(k, i) = (k^2 + i) \bmod m$ is an example of linear probing.

   True False In a binary search tree where all nodes have distinct keys, the node with the minimum key can't have a left child.

2. (6 points) Fill out this table. Assume that all elements in the array are distinct (except when specified otherwise) positive integers between 1 and k. Runtimes should be expressed in big-Θ notation, in terms of n and k. Assume that Radix Sort uses Counting Sort for each digit.

	Runtime when array already sorted in increasing order	Runtime when array already sorted in decreasing order	Runtime when all elements in array are the same	Runtime for randomly distributed array	In place?	Stable?
Insertion Sort						
Merge Sort						
Quicksort						
Heapsort						
Counting Sort						
Radix Sort						

3. (5 points) In the hash table below, show the result of inserting the keys 7, 1, and 8 using open addressing, with both the linear probing function $h(k,i) = (k + 3i) \bmod 6$, and the quadratic probing function $h(k,i) = (k + i^2) \bmod 6$.

   Then, suppose that the next value to be inserted (after 7, 1, and 8) will be a randomly chosen integer x in the range 100 to 699, inclusive. Compute the probability, as either a fraction or a percentage, that x will be placed into each of the slots in the table for each probing function. You must also compute the probability that x will not be able to be inserted into the table no matter what i value is chosen.

Slot	Key (linear) $h(k,i)=(k+3i) \bmod 6$	Key (quadratic) $h(k,i)=(k+i^2) \bmod 6$	Probability that x will be inserted here (linear)	Probability that x will be inserted here (quadratic)
0				
1				
2				
3				
4				
5				
Can't be inserted	■	■		

4. (4 points) Why do we need a Build-Max-Heap function to initialize the max heap at the beginning of Heapsort, when we could accomplish the same goal by just calling Max-Heap-Insert n times?