Question 1:

1.1

- The worst case is $\Theta(n^2)$, which means we always choose the biggest or smallest element and can only eliminate one element for each recursion round.

1.2

- The expected runtime is $O(nlogn)$, if we successfully choose the median everytime, then we can eliminate half of the remaining elements for each recursion round and this would be the ideal case.

1.3

- In order to shorten the runtime, I can select the median element as the pivot for each recursion. Median finding is $O(n)$ and the worst case for this quicksort would become $O(nlogn)$.

1.4

- example: $[3_1, 1_1, 2, 3_2, 1_2]$, by using QuickSort we swap non-adjacent elements, and this becomes: $[1_2, 1_1, 2, 3_2, 3_1]$, the original order is gone.

Question 2:

- Runtime for Insertion is longer, because after we find the correct linkedlist we also need to go through it to find the right spot to insert the next element, so total runtime is $O(1 + \alpha + n)$
- Deletion is not changed, still $\Theta(1)$
- Successful search is also not changed: $\Theta(1 + \alpha)$
- Unsuccessful search is: $\Theta(1 + \alpha)$, but when the data is very big, the runtime can be faster since we can say that an element is impossible to be inside certain linkedlist (for example a sorted linkedlist with smaller element on the front, if desired element is smaller than the first element, then we do not need to go into this linkedlist at all)

Question 3:

index	linear probing	quadratic probing	double hashing
0	22	22	22
1	88		
2		88	59
3		17	17
4	4	4	4
5	15		15
6	28	28	28
7	17	59	88
8	59	15	
9	31	31	31
10	10	10	10

- For linear probing:

h(k,i)=(k+i) mod 11	T0	T1	T2	T3	T4	T5	T6	T7	T8
0 mod 11		22	22	22	22	22	22	22	22
1 mod 11								88	88
2 mod 11									
3 mod 11									
4 mod 11				4	4	4	4	4	4
5 mod 11					15	15	15	15	15
6 mod 11						28	28	28	28
7 mod 11							17	17	17
8 mod 11									59
9 mod 11			31	31	31	31	31	31	31
10 mod 11	10	10	10	10	10	10	10	10	10

- For quadradic probing:

h(k,i)=(k+i+3i^2) mod 11	T0	T1	T2	T3	T4	T5	T6	T7	T8
0 mod 11		22	22	22	22	22	22	22	22
1 mod 11									
2 mod 11								88	88
3 mod 11							17	17	17
4 mod 11				4	4	4	4	4	4
5 mod 11									
6 mod 11						28	28	28	28
7 mod 11									59
8 mod 11					15	15	15	15	15
9 mod 11			31	31	31	31	31	31	31
10 mod 11	10	10	10	10	10	10	10	10	10

- For double hashing:

h(k,i)=(k+i(1+k mod 10)) mod 11	T0	T1	T2	T3	T4	T5	T6	T7	T8
0 mod 11		22	22	22	22	22	22	22	22
1 mod 11									
2 mod 11									59
3 mod 11							17	17	17
4 mod 11				4	4	4	4	4	4
5 mod 11					15	15	15	15	15
6 mod 11						28	28	28	28
7 mod 11								88	88
8 mod 11									
9 mod 11			31	31	31	31	31	31	31
10 mod 11	10	10	10	10	10	10	10	10	10