Answer to Module 3 Problem Set

- Student Name: Gaole Dai
- Student ID: S01435587

Problem 1

Selection Sort:

Times	Last Step	Trace
Original	/	T H I S Q U E S T I O N
First Sort	T H I S Q U E S T I O N	E H I S Q U T S T I O N
Second Sort	**E** H I S Q U T S T I O N	E H I S Q U T S T I O N
Third Sort	**E H** I S Q U T S T I O N	E H I S Q U T S T I O N
Fourth Sort	**E H I** S Q U T S T I O N	E H I I Q U T S T S O N
Fifth Sort	**E H I I** Q U T S T S O N	E H I I N U T S T S O Q
Sixth Sort	**E H I I N** U T S T S O Q	E H I I N O T S T S U Q
Seventh Sort	**E H I I N O** T S T S U Q	E H I I N O Q S T S U T
Eighth Sort	**E H I I N O Q** S T S U T	E H I I N O Q S T S U T
Ninth Sort	**E H I I N O Q S** T S U T	E H I I N O Q S S T U T
Tenth Sort	**E H I I N O Q S S** T U T	E H I I N O Q S S T U T
Eleventh Sort	**E H I I N O Q S S T** U T	E H I I N O Q S S T T U
Final	**E H I I N O Q S S T T U**	E H I I N O Q S S T T U

Problem 2

Insertion Sort:

Times	Last Step	Trace
Original	/	T H I S Q U E S T I O N
First Sort	T **H** I S Q U E S T I O N	H T I S Q U E S T I O N
Second Sort	H T **I** S Q U E S T I O N	H I T S Q U E S T I O N
Third Sort	H I T **S** Q U E S T I O N	H I S T Q U E S T I O N
Fourth Sort	H I S T **Q** U E S T I O N	H I Q S T U E S T I O N
Fifth Sort	H I Q S T **U** E S T I O N	H I Q S T U E S T I O N

Times	Last Step	Trace
Sixth Sort	H I Q S T U **E** S T I O N	E H I Q S T U S T I O N
Seventh Sort	E H I Q S T U **S** T I O N	E H I Q S S T U T I O N
Eighth Sort	E H I Q S S T U **T** I O N	E H I Q S S T T U I O N
Ninth Sort	E H I Q S S T T U **I** O N	E H I I Q S S T T U O N
Tenth Sort	E H I I Q S S T T U **O** N	E H I I O Q S S T T U N
Eleventh Sort	E H I I O Q S S T T U **N**	E H I I N O Q S S T T U
Final	E H I I N O Q S S T T U	**E H I I N O Q S S T T U**

Problem 3

M U C H L O N G E R Q U E S T I O N

Times	Last Step	Trace	h-seq
Original	/	M U C H L O N G E R Q U E S T I O N	
1	M U C H L O N G E R Q U E S T I O N	I U C H L M N G E R O U E S T Q O N	5
2	I U C H L M N G E R O U E S T Q O N	I N C H L M O G E R O U E S T Q U N	5
3	I N C H L M O G E R O U E S T Q U N	I N C H L M O E E R O U G S T Q U N	5
4	I N C H L M O E E R O U G S T Q U N	I N C E L M O E H R O U G S T Q U N	5
5	I N C E L M O E H R O U G S T Q U N	I N C E L M O E H R O U G S T Q U N	5
6	I N C E L M O E H R O U G S T Q U N	I N C E L M O E H R O U G S T Q U N	5
7	I N C E L M O E H R O U G S T Q U N	I N C E L M O E H R O U G S T Q U N	5
8	I N C E L M O E H R O U G S T Q U N	I N C E L M O E H R O U G S T Q U N	5
9	I N C E L M O E H R O U G S T Q U N	I N C E L M O E H R O U G S T Q U N	5
10	I N C E L M O E H R O U G S T Q U N	I N C E L M O E H R O U G S T Q U N	5

Times	Last Step	Trace	h-seq
11	I N C E L M O E H R O U G S T Q U N	I N C E L M O E H R O U G S T Q U N	5
12	I N C E L M O E H R O U G S T Q U N	I N C E L M O E H R O U G S T Q U N	5
13	I N C E L M O E H R O U G S T Q U N	I N C E L M O E H R O U G S T Q U N	5
14	I N C E L M O E H R O U G S T Q U N	C N G E H M I E L R O U O S T Q U N	2
15	C N G E H M I E L R O U O S T Q U N	C E G E H M I N L N O Q O R T S U U	2
16	C E G E H M I N L N O Q O R T S U U	C E G E H M I N L N O Q O R T S U U	2
17	C E G E H M I N L N O Q O R T S U U	C E G E H M I N L N O Q O R T S U U	2
18	C E G E H M I N L N O Q O R T S U U	C E G E H M I N L N O Q O R T S U U	2
19	C E G E H M I N L N O Q O R T S U U	C E G E H M I N L N O Q O R T S U U	2
20	C E G E H M I N L N O Q O R T S U U	C E G E H M I N L N O Q O R T S U U	2
21	C E G E H M I N L N O Q O R T S U U	C E G E H M I N L N O Q O R T S U U	2
22	C E G E H M I N L N O Q O R T S U U	C E G E H M I N L N O Q O R T S U U	2
23	C E G E H M I N L N O Q O R T S U U	C E G E H M I N L N O Q O R T S U U	2
24	C E G E H M I N L N O Q O R T S U U	C E G E H M I N L N O Q O R T S U U	2
25	C E G E H M I N L N O Q O R T S U U	C E G E H M I N L N O Q O R T S U U	2
26	C E G E H M I N L N O Q O R T S U U	C E G E H M I N L N O Q O R T S U U	2
27	C E G E H M I N L N O Q O R T S U U	C E G E H M I N L N O Q O R T S U U	2
28	C E G E H M I N L N O Q O R T S U U	C E G E H M I N L N O Q O R T S U U	2

Times	Last Step	Trace	h-seq
29	C E G E H M I N L N O Q O R T S U U	C E G E H M I N L N O Q O R T S U U	2
30	C E G E H M I N L N O Q O R T S U U	C E E G H I I L N N O O O R S T U U	1

Problem 4

Yes.

- For selection sort, it always picked the highest or lowest element from the unsorted part and putting it at the beginning. Thus, whenever all the keys are identical, the complexity for making **comparison** is $n-1+n-2+n-3+\ldots+1=\mathcal{O}(n^2)$, and the complexity for **exchange** is $\mathcal{O}(m)$.
- For insertion sort, its complexity of making **comparison** is the same as selection sort. However, since all the keys are identical, it did **not need to make any exchange**.
- Thus, if all keys are identical, insertion sort work better than selection sort.

Problem 5

No.

- For selection sort, as problem 4 stated, its **comparison** is always $n-1+n-2+n-3+\ldots+1=\mathcal{O}(n^2)$, and the complexity for **exchange** is $\mathcal{O}(m)$.
- For insertion sort, the **comparison** is the same as selection sort. However, since all keys are in reverse order, the complexity for keys exchange is at worst $n-1+n-2+n-3+\ldots+1=\mathcal{O}(m^2)$. The total cost of comparison and exchange for insertion sort is worse than selection sort.
- Thus, if all keys are in reverse order, selection sort work better than insertion sort.

Problem 6

Steps:

1. Sort the list
2. Iterate through the sorted list, check and eliminating duplicates by looking the item next to each other. The cost is $\mathcal{O}(n)$.

Python code is shown below:

```
def removeDuplicates(inputList):
    inputList.sort()
    i = 0
    while i < len(inputList)-1:
        if (inputList[i] == inputList[i+1]):
            inputList.remove(inputList[i+1])
        else:
            i += 1
    return inputList
```

Problem 7

Steps:

1. Sort letters of each string.
    1. String a, sting b: a'= sort string a'; b'=sort string b'; if a == b then a and b originally are jumbles
2. Sort letter, then sort the **signature** to put jumbles together. The complexity for printing out the result is $\mathcal{O}(N)$.

Python code is shown below:

```
def jumbleWords(inputList):
    # Sort the string elements in the input list
    res = [''.join(sorted(element)) for element in inputList]
    # Join the key and sorted value into signature
    tuple_list = list(zip(res, inputList))
    # Sort the tuple to put jumbles together
    tuple_list.sort(key = lambda x: x[0])
    # Print out the jubmle words
    prev = tuple_list[0][0]
    for i, (a, b) in enumerate(tuple_list):
        if (a == prev):
            print(b, end =" ")
        else:
            print('\n')
            print(b, end =" ")
        prev = a
```

Problem 8

Problem 9

- In overall, insertion sort works better than selection sort.
- Insertion sort is faster on inputs that are **partially-sorted**. If array is partially-sorted, the cost for insertion sort is at best $\mathcal{O}(n)$, which indicates that insertion sort make n comparison (scenario also includes Problem 4).
- But the cost for selection sort always take $\mathcal{O}(n^2)$ for worst case and average case.