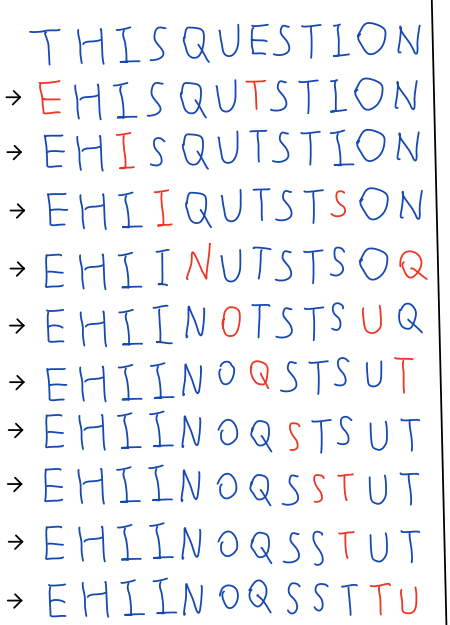
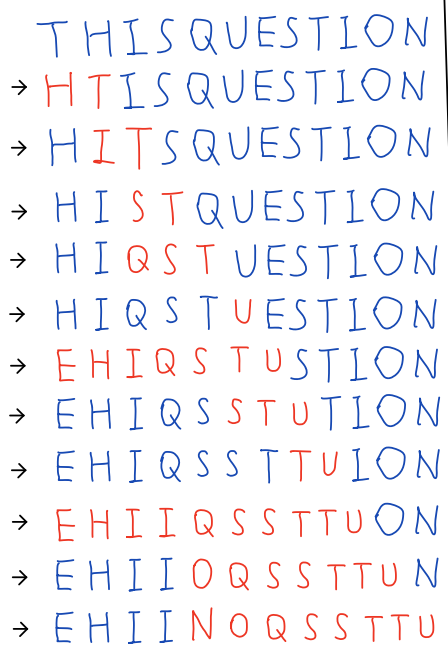
Module 3 Problem Set Hsuan-You(Shaun) Lin

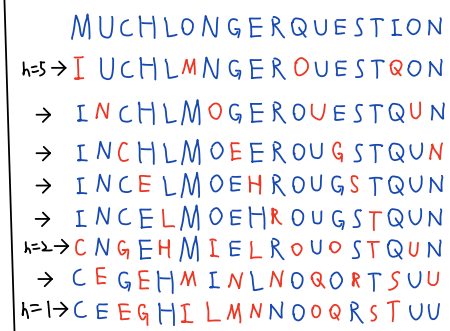
1. My Answer: (selection sort)



2. My Answer: (insertion sort)



3. My Answer: (shell sort- 5,2,1)



4. My Answer: Yes, if all keys are identical, insertion sort definitely work better than selection sort.
My reason:

Insertion sort has to sweep the sorted list each time to determine where to insert each element, and have best case of O(N), but selection sort works by taking the smallest element in an unsorted array and bring it to the front, it's only have O(N^2) best case, apparently insertion sort work better than selection sort.

5. My Answer: No, if all keys are in reverse order, insertion sort does not work better than selection sort.
My reason:

Because both sorting algorithms compare the same amount of pairs, but insertion sort exchange more than selection sort. In conclusion, selection sort work better than insertion sort.

6. My Answer: In python, we can use set() constructor, pass the list as argument to the set constructor, and it returns a set of unique items. If we can't use set() constructor, we can also use for loop to iterate over the elements of the list, and check if the element has occurred only once, than choose a sorting algorithm to sort the results.

7. My Answer: Figure1 is my Python code

```python
from collections import defaultdict as DF
import bisect # If you want to use insertion sort, you can use bisect.insert()

# Creat Jumble hash-map to put sorted words
Jumble = DF(list)

def GroupJumbles(Words):
    for i in Words:
        # Take letters from list words and sorting
        Letter = str(sorted(i))
        Jumble[Letter].append(i)

    for Letter in Jumble:
        print("Jumbles:")
        print(" ".join(Jumble[Letter]))

Words = ["racing", "secura","saucer", "caring", "random"]
GroupJumbles(Words)
```

Ln: 18, Col: 20

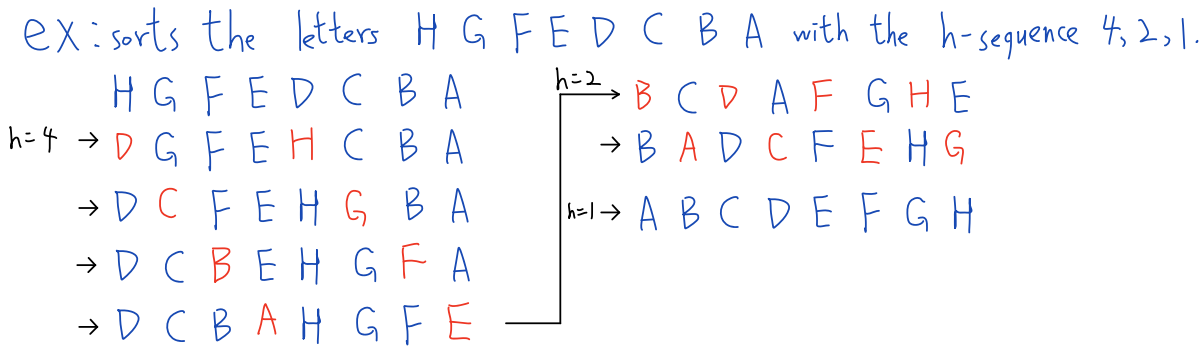
Run | Share | Command Line Arguments

```
Jumbles:
racing caring
Jumbles:
secura saucer
Jumbles:
random


** Process exited - Return Code: 0 **
Press Enter to exit terminal
```

Figure 1.

8. My Answer: Because in the h-sequence 1, 2, 4, 8, ..., 2^k, 2k is redundant in the sorting, we can just easily sort k, and k-length sequence will cover all 2k-length position, the example below will demonstrate the worst case.



9. My Answer: Because we won't get better performance to use selection sort for h-sequence in the shell sort. For example, if we use k-sequence for the shell sort and a list of N length, then the complexity will becomes O(kN^2), which is worse then the best case of selection sort O(N^2). In conclusion, the reason above explained that why not to use selection sort for h-sequence in the shell sort.