Linear Search Tasks

Exercise 1.1 – linear11.py

Implementing linear search in Python

Implement the linear search algorithm in Python. Here is the pseudo code:

```
index = 0
while index < length of
array if A[index] equals
target  return index index =
index + 1 return not found
```

You could complete the following outline, which is followed by two test cases:

```
def findLin(A, target):

# find the target in array A
# return index or -1
 ...
print(findLin([2,3,4],3)
print(findLin([2,3,4],7)
```

Binary Search Tasks

Exercise 1.2

Complete the iterative and recursive binary search procedures in binarysearch.py

Exercise 1.4

- Using playing cards and a pointer (or pointers), show how the following search algorithms work
  - Linear search
  - Binary search
- A pen or pencil can be used as a pointer
- Do we need a pointer?

Bubble Sort

Exercise 2.1: Sort Sequence

The following array of numbers is sorted using bubble sort. Show the array after each pass, starting from the left hand side:

56	21	43	61	12	37	13

Exercise 2.2: Coding a bubble sort

Bubbletask.py

```
def bubblesort(alist):
    #find the length of the list n
    # loop n times
        # loop n-1 times
            #if the current item is bigger than the next item
                #swap them
        # print to show each pass through the list?
```

Exercise 2.3

Insertion Sort

Exercise 3.1: Sort Sequence

Repeat Exercise 2.1 for insertion sort. Each row should show the order after each insertion.

Note that it is also possible to show the order after each swap operation (in the example given, this requires more rows).

Show the array after each pass, starting from the left hand side:

56	21	43	61	12	37	13

Exercise 3.2: Demonstration Insertion Sort

Repeat Exercise 2.2 for insertion sort. Again, this can be done in more or less detail:

- In the less detailed version, use one pointer to point at the card that is being inserted. Start with the second last card. Treat the insertion as one operation.

- In the more detailed version, you need two pointers: ○ The first pointer points to the card that is to be inserted (as above) but does not move during the insertion.
    - ○ The second pointer is used to keep track of the insertion: it moves as the card is inserted. At each step, compare the value of the card being inserted with the card to its right.

Discuss which version is better.

Exercise 3.3: Complete insertionsort.py

The insertionsort.py contains an incomplete insertion sort procedure. Complete this.

Quicksort

Exercise 2.6: Demonstrate Quicksort

Demonstrate Quicksort in the style of Exercises 2.4 and 2.2. You are recommended not to show the partition step in too much detail.