Tutorial 04 & 05
Sort

1. The Bubble Sort algorithm can be optimized by checking if there are any swapping operation in the inner loop. If there is no swapping in any pass, it means that the list is already fully sorted, and we can stop the sorting process. In other words, we can optimize the number of passes required when the list gets sorted before the completion of all passes.

The following code is an incomplete implementation of this optimized Bubble Sort described above:

```
# Sorts a sequence in ascending order using the
# optimized bubble sort algorithm
def bubbleSort_optimized( theSeq ):
    n = len( theSeq )

    # Perform n-1 bubble operations on the sequence
    for i in range(n - 1, 0, -1):
        # Set boolean variable to check occurrence of swapping
        # in inner loop
        ____________________

        # Bubble the largest item to the end
        for j in range(i):
            if theSeq[j] > theSeq[j + 1]:
                # Swap the j and j+1 items
                tmp = theSeq[j]
                theSeq[j] = theSeq[j + 1]
                theSeq[j + 1] = tmp

                # Set boolean variable value if swapping occurred
                ____________________


        # Exit the loop if no swapping occurred
        # in the previous pass

        ____________________
            ____________________
```

a. Complete the implementation by providing the rest of the required code for the optimized Bubble Sort.

b. How many pass(es) is(are) required to sort the following list of numbers with the standard Bubble Sort and the optimized Bubble Sort?

   - [1, 2, 3, 5, 4]

c. How many pass(es) is(are) required to sort a list of numbers that are already sorted with the optimized Bubble Sort?

2. Given the following list of numbers:
   - 12, 7, 9, 24, 7, 29, 5, 3, 11, 7

   Using the table format shown below, trace each pass of the sorting process.

Pass	12	7	9	24	7	29	5	3	11	7
1										
2										
…										

   a. Use the Selection Sort algorithm to sort the above list in descending order.

   b. Use the Insertion Sort algorithm to sort the above list in ascending order.

3. State whether each of the following statements are True or False:

   a. Using Selection Sort, after **N** passes through the list, the first **N** elements are in sorted order.

   b. Using Selection Sort (ascending order), after **N** passes through the list, the first **N** elements are the **N** smallest elements in the list.

   c. Using Insertion Sort, after **N** passes through the list, the first **N** elements are in sorted order.

   d. Using Insertion Sort (ascending order), after **N** passes through the list, the first **N** elements are the **N** smallest elements in the list.

-- End of Tutorial --