MH1403 Algorithms and Computing
Lab 4 Sorting and Divide-and-Conquer Algorithms

Submission Instructions:

1. This lab is 5% of the final grade of this course.
2. The submission deadline is 11:59PM, 19 April (Sunday).
3. You need to submit the codes of Task 1 and Task 2 through NTULearn.

Task 1. (2.5 marks) Write a Python function insertionSort() to sort a list (we assume that all the elements in the list are numerical values). The input to this function is a list to be sorted. You should not use the sort() function of Python in this task. Note that you do not print anything from the function insertionSort.

Create a list containing the elements (5*i+2)%37 for i from 0 to 20. Call the function insertionSort() to sort this list, print the sorted list to the screen.

You write the code in the file task1.py

Task 2. (2.5 marks) Write a Python function mergeSort() to sort a list (we assume that all the elements in the list are numerical values). The input to this function is a list to be sorted. You should not use the sort() function of Python in this task. Note that you do not print anything from the function mergeSort.

Create a list containing the elements (5*i+2)%37 for i from 0 to 20. Call the function mergeSort() to sort this list, print the sorted list to the screen.

You submit your code in the file task2.py

Task 3. (optional) When we learn the maximum subsequence problem in Lecture 1 (the last 7 slides of Lecture 1), we mentioned that the maximum subsequence problem can be solved using the divide-and-conquer algorithm with very low running time.

The divide-and-conquer algorithm on the maximum subsequence problem works as follows:

- Locate the middle element of the sequence. It divides the sequences into three parts: the left half, the right half, and the middle element.
- Find the sum of the maximum subsequence for the left half (in a recursive way)
- Find the sum of the maximum subsequence for the right half (in a recursive way)
- Find the sum of the maximum sequence that contains the middle element (how to solve this problem in an efficient way?)
- Return the maximum of the above three sums.

Write a python function maximumSequence() that computes the sum of the maximum sequence. The input to this function is a list.

Create a list A as [4, –3, 5, –2, -1, 2, 6, -2]. Call the function maximumSequence() to compute the sum of the maximum sequence of List A, print the sum to the screen.