CE2101/CZ2101
ALGORITHM DESIGN AND ANALYSIS

Project 1: Integration of InsertionSort & MergeSort

In MergeSort, when the sizes of subarrays are small, the overhead of many recursive calls makes the algorithm inefficient. Therefore, in real use, we often combine MergeSort with InsertionSort to come up with a hybrid sorting algorithm for better efficiency. The idea is to set a small integer **S** as a threshold for the size of subarrays. Once the size of a subarray in a recursive call of MergeSort is less than or equal to **S**, the algorithm will switch to InsertionSort, which is efficient for small input.

(a) Implement the above hybrid algorithm. Analyze its time complexity in terms of the number of key comparisons with respect to **S** and the size of the input list n. Study how to determine an optimal value of **S** for best performance of this hybrid algorithm on different input cases and input sizes.

(b) Implement the original version of MergeSort (as learnt in lecture). Compare its performance against the above hybrid algorithm in terms of the number of key comparisons and CPU time on different input cases and input sizes. You can use the optimal value of **S** obtained in (a) for this task.

Show your running program with printings of CPU time, and present your analysis.