CSCI 1112 Algorithms and Data Structures

Lab 6 – More Recursion

Part 1: Merge Sort (15 points)

a) Download MergeSort.java

b) Implement the *merge*() method used in *mergeSort*(). Count the number of element-wise comparisons and the number of copy operations. Report the performance in the table below and compare with the performance of insertion sort.

c) Merge sort can be improved if the algorithm stops before the merge step when the elements of the array are already sorted. If the right-most element in the first subarray is smaller than or equal to the left-most element in the second subarray, then the array is already sorted and there is no need to merge. Implement this step and report the change in counts.

Algorithm	N=10		N=100		N=1000	
	compare	Swap/copy	compare	Swap/copy	compare	Swap/copy
Insertion sort	9	9	99	99	999	999
Merge sort [2.b]	56	34	1225	672	18669	9976
Merge sort [2.c]	46	22	765	159	13623	4518

Part 2: Recursion (5 points)

a) Write a recursive method that takes a string of letters and returns all the n! permutations of the string. For example, if the string is “abc”, the method should return the following strings:

abc

acb

bac

bca

cab

cba