Problem 1. Assume that your computer has special hardware that finds the minimum of k (or fewer) elements in one comparison step. Your answers to this question should have n and k as parameters.

(a) Design an efficient algorithm based on Merge Sort to sort n elements using this special hardware. (This is an upper bound.)

(b) Analyze your algorithm. Get the high order term exactly.

(c) Use decision trees to find a lower bound for sorting when using this special hardware.

(d) Compare your upper and lower bounds.

Problem 2. To sort an array, $A = [6, 0, 2, 0, 1, 3, 4, 6, 1, 3, 2]$, suppose we rewrite the last **for** loop header in the version 2 of the counting sort algorithm covered in class as,

```
for j = 1 to n do
```

Show that the algorithm still works properly. Is the modified algorithm stable?

Problem 3.

(a) Assume you have an alphabet of letters from **"e"** to **"u"**. Illustrate the operation of radix sort on the following list of English words:

`tote, soup, soot, pout, toot, sups, tour, opts, rout, tors`

(b) Use the words **"sup"** and **"tor"** in one English sentence that shows that you understand the meaning of both words.