NAMES: Dakota Krogmeier, Kristopher Hoce, John Karuntzos

Worksheet 20: Work with either ONE or TWO other students to complete this worksheet. You must work with at least one other student on this assignment, and should submit one paper.

1. Develop an efficient algorithm to carry out the following sorting approach (4 points):

   KEY IDEA: An array A consists of n numbers. It will be sorted by first finding the largest element of A and exchanging it with the element in A[n]. Then it will find the second largest element of A, and exchange it with A[n-1]. Continue in this manner for the largest n – 1 elements of A.

   Algorithm SelectionSort(*A*):

   Input: Array A consisting of n integers

   Output: Array A sorted in ascending order

   PROCESS:

```
For i <- n to 1:
    max <- 0
    For x <- 1 to i
        If A[x] > A[max]:
            Max <- x
    Temp <- A[x]
    A[x] <-A[i]
    A[i] <-temp
Return A
```

2. Re-write the merge algorithm described in class so that it stops once either L or R has had all of its elements copied back to A and then copies the remaining elements of the other array back into A. Your algorithm should not extend the L or R arrays to have ∞ as the last element in the array as a result of your modification. (2 points)

```
Merge(L, R):
    A = [ ]
    Size s1 of L
    Size s2 of R
    Int i, j, k = 0;  //k iterates through list A
    While (i < s1 AND j < s2):  // if index becomes greater than size of array, fail condition
        If(L[i] < R[j]):
            Add L[i] to A[k]
            Increment i, k
        Else
            Add R[j] to A[k]
            Increment j, k
    //If first while loop condition fails, add remaining elements of array.
    While (i < s1):
        A[k++] <- L[i++]
    While (j < s2):
        A[k++] <- R[j++]
Return A;
```