Assignment 2. **Analysis of loop algorithms, applications of sorting.**

Given January 29, due February 5.

1. We have two sets of objects, $A$ and $B$, each with $n$ objects. We have a table $W(i,j) = \text{dist}(i,j)$ that gives the (presumably precomputed) distances between object $i$ in $A$ and object $j$ in $B$. We do not have, for example, the distance between object 1 and object 2 in $A$. Rather, $W(1,2)$ is the distance between the first object in $A$ and the second object in $B$. We want to match objects in $A$ with nearby objects in $B$. Our algorithm will use the following single match step. One single match step of sets $A$ and $B$ is to find the closest pair of objects in $A$ and $B$

$$\min_{i \in A, j \in B} W(i,j).$$

We then remove the closest pair, one from $A$ and one from $B$, leaving $A$ and $B$ with $n - 1$ elements each. The total match process is to do the single match operation $n$ times until every object from the original $A$ is matched with some object from the original $B$.

**a.** Write pseudocode to describe this algorithm, assuming that $A$ and $B$ are implemented as arrays. Use a procedure `singleMatch(A,B,W,k,n)` that does the single match. Be careful to remove the matched elements from $A$ and $B$ so that on returning $A$ and $B$ have one fewer elements than before. Implement $A$ and $B$ as arrays of integers representing the places left at that stage. At the beginning $A = (1,2,3,...,n)$ and the same for $B$. Here $k$ is the number of items left in $A$ and $B$ and $n$ is the number originally, which is used for the dimension of the doubly indexed array, $W$.

**b.** What is the running time for your algorithm?

2. What is the asymptotic time of the following algorithm:

```c
for k = 1, ... , n {
    for j = 1, ... , k {
        a(j) = rand();
    }
    mergesort(a,k,temp){
        sum = 0;
        for j = 1, ... , k-1 {
            sum = sum + ( a(j+1) - a(j) ) * ( a(j+1) - a(j) );
        }
    }
    cout << " the sum is " << sum << endl; // Never mind this line.
```
The routine `rand()` produces a different random number in between 0 and 1 each time it is called.

3. We have a sorted list of real $n$ numbers, $a(1) > a(2) > \cdots > a(n)$. For any number, $x$, $h(x)$ is the number of elements of $a$ larger than $x$. Suggest an algorithm that will find the largest element of $a$ that is smaller than $x$ in time of order $\log(h(x))$.

4. We have a sorted list of $n$ real numbers, $a(1) > a(2) > \cdots > a(n)$. Give an $O(n)$ algorithm that determines whether $a(j) = -a(i)$ for any pair $i, j$. 