CS2223
HW#2

DUE: Monday, November 8

1. (9 points) Do Problem 2-4, parts a, b and c on pages 39→40 of our text.

2. (5 points) Do Exercise 2.2-2 on pg. 27 of our text.

3. (5 points) Describe an algorithm to sort \( n > 1 \) numbers such that the best-case running time is in \( O(n) \).

4. (7 points) Assuming that \( n \) is a power of 2, what value is returned by the following algorithm? Give an exact answer; do not use asymptotic notation.

```plaintext
k ← n
sum ← 0
while \( k \geq 1 \) do
    for \( i \leftarrow 1 \) to \( k \) do
        sum ← sum + 1
    k ← k / 2
return sum
```
1. **a** $(1,5), (2,5), (3,5), (4,5)$ and $(3,4)$

**b** Array $(n, n-1,...,2,1)$ has $\frac{n(n-1)}{2}$ inversions.

**c** All the statements of INSERTSORT are executed a linear number of times, $\Theta(n)$, except statements 6 and 7. They are executed once for each inversion. So the execution time for INSERTIONSORT on input array $A$ is $\Theta(n + \gamma(A))$, where $\gamma(A)$ is the number of inversions in $A$.

2. The invariant is that after the $i^{th}$ smallest element of $A$ is exchanged with $A[i]$, 

- the $i$ smallest elements of $A$ are in $A[1..i]$, and they are sorted.

If the $n-1$ smallest elements of $A$ are in $A[1..n-1]$, then the $n^{th}$ smallest element of $A$, which must be the largest element of $A$, must be in $A[n]$. As shown in class, this algorithm always takes time in $\Theta(n^2)$.

3 In linear time, check if $A$ is sorted. If it’s not sorted, then INSERTIONSORT it.

```plaintext
sorted? ← true
for i ← 1 to n-1 do
    if A[i] > A[i+1] then sorted? ← false
if not sorted? then INSERTIONSORT(A)
```

4. After the 1st time through the for-loop, when $k=n$, sum is set to $n$. After the 2nd time through the for-loop, when $k=n/2$, sum is set to $3n/2$. After the $j^{th}$ time through the loop, when $k = \frac{n}{2^{j-1}}$, sum is $n + \frac{n}{2} + \ldots + \frac{n}{2^{j-1}}$. This summation can be rewritten as

$$
\sum_{i=0}^{j} \frac{n}{2^{i+1}} = n \sum_{i=0}^{j} \left(\frac{1}{2}\right)^{i+1}.
$$

Replacing $i$ by $i+1$ yields $n \sum_{i=0}^{j} \left(\frac{1}{2}\right)^{i+1} = n \sum_{i=0}^{j} \left(\frac{1}{2}\right)^{i} = n \frac{1 - \left(\frac{1}{2}\right)^{j+1}}{1 - \frac{1}{2}} = 2n \left(1 - 2^{-j}\right)$.

The while-loop executes $\lg n + 1$ times, so the algorithm returns

$$
2n \left(1 - \frac{1}{2^{\lg n+1}}\right) = 2n \left(1 - \frac{1}{2n}\right) = 2n - 1.
$$