CSE 373 – Data Structures and Algorithms
Autumn 2003
Dry assignment #1.
Due date: 10/10/03 (see submission instructions in course web-page).

1. (12 points) Explain in at most two sentences what is wrong, if anything, with the following induction proof.

- Claim: All horses are the same color.
- Proof: We prove this by showing that any set of horses contains only horses of a single color; in particular, this is true for the set of all horses. Let H be an arbitrary set of horses. We show by induction on n, the number of horses in H, that all horses in H are the same color.
- Basis: The cases n = 0 and n = 1 are immediately seen to be true.
- Induction step: Consider any number n of horses in H. Call these horses h_1, h_2, h_3, ..., h_n. By the induction hypothesis, any set of n-1 horses contains only horses of a single color. Consider the set H_1 obtained by removing horse h_1 from H, and the set H_2 obtained by removing horse h_2 from H. There are n-1 horses in each of these sets, hence the induction hypothesis applies to each: H_1 has all horses of a single color (say, c1), and H_2 has all horses of a single color (say, c2). But, since horse h_n is common to *both* sets H_1 and H_2, the two colors c1 and c2 must be the same. This completes the induction step.

2. (24 points) True or False? Give a brief explanation.

a. $\sum_{k=1}^{n} k = O(n)$

b. $\sum_{k=1}^{n} k = \Omega(n)$

c. $2^n = \Theta(3^n)$

d. $3n^2 + n + n \cdot \log(n) = \Omega(n^2)$

e. $3n^2 + n + n \cdot \log(n) = \Omega(n \cdot \log(n))$

f. $\frac{n^2}{2^n} = O(1)$

3. (20 points) For each of the following questions, briefly explain your answer.

a. If I prove that an algorithm takes $O(n^2)$ worst-case time, is it possible that it takes $O(n)$ on some inputs?
b. If I prove that an algorithm takes $O(n^2)$ worst-case time, is it possible that it takes $O(n)$ on all inputs?
c. If I prove that an algorithm takes $\Theta(n^2)$ worst-case time, is it possible that it takes $O(n)$ on some inputs?
d. If I prove that an algorithm takes $\Theta(n^2)$ worst-case time, is it possible that it takes $O(n)$ on all inputs?

4. (24 points) The sequence 0,1,1,3,5,11,21,43 is given by
$S_0=0$, $S_1=1$.
$S_k=S_{k-1}+ 2*S_{k-2}$ (k>1)
Write (in pseudocode) a function that gets as input a **sorted** array a[] and its length n and returns the maximal index k such that all the numbers $S_0,S_1,...,S_k$ appear in a[] (or –1 if such an index does not exist).
The time complexity of your function should be O(n). The space complexity should be O(1).

Examples: For a={0, 1, 1, 2, 3, 4, 5, 11, 15, 17, 21, 56, 67} the returned value should be 6 (S_0, S_1,..., S_6 are in the array).
For a={-5, -2, 0, 1, 1, 1, 2, 3, 3, 4, 5, 5} the returned value should be 4
For a={–8,1,2,3,4,5} the returned value should be -1

5. (20 points) Write (in pseudocode) a recursive function 'MaxPair' that gets an array a[] of integers and its size n (it is known that n>1), and returns the maximal sum of two consecutive elements in a[] (that is $Max(a[j-1]+a[j] : 1 \leq j \leq n-1)$). You are not allowed to use loops in your solution.
What is the time and space complexity?