CSCI 385 Data Structures and Algorithms
Test 1
Fall 2003

- Please answer all questions carefully and thoughtfully.
- Please number each question.
- Please place your name on each page.
- Any answer, other than a definition, should an indication on how you obtained your answer, or a justification for the answer given.

1. (3 points) State the definition for $O(f(n_)$, (ie $f(x) \in O(g(x))$ if ...)
2. (4 points) State at least two advantages of using asymptotic analysis to measure the performance of an algorithm.
3. (5 points) If algorithm A is $O(f(n))$ and algorithm B is $O(g(n))$ and $f(n) < g(n)$ (for example $f(n) = n^2$ and $g(n) = n^n$), is it always desire able to use algorithm A? State why or give a counter example.
4. (3 points) You are asked to review a paper for Professor Smith where he claims that he can, using comparisons only, determine the i^{th} largest item in an array of n items in $O(lg(lg(n)))$ time (worst case). Do you advise Prof Smith to publish his work or to reexamine his work carefully to find an error in his analysis? Why?

   Hint: Can you devise a sort to use this algorithm using his selection algorithm?
5. (2 points) Show that 2^{100} is $O(1)$.
6. (3 points) Show that $5n$ is $O(n)$.

Please turn the test over for more questions.

7. The insertion sort:

```
(0) InsertSort(array,n)
(1)    for i = 2 to n
(2)       tmp = array[i]
(3)       j = i;
(4)       while ((j>1) and (array[j-1] > tmp))  do
(5)           array[j] = array[j-1]
(6)           j = j - 1
(7)       end while
(8)       array[j] = tmp
(9)    end for
```

   (a) (2 points) Trace the algorithm for the array 4,3,7,2,1

   (b) Using loop invariants, argue that the algorithm will produced a sorted array

   (c) (2 points) Does this algorithm sort in place? (Why or Why not?)

   (d) (2 points) Is this algorithm a stable sort? (Why or why not?)

   (e) (4 points) What are the best and worst cases for this algorithm? (Argue this)

8. (5 points) If an array is sorted, the binary search algorithm will determine if an item (element) is in the array is as follows:

```
(1)   bool BinarySearch(array , start,  end, element)
(2)       if start >= end then return false
(3)       mid = start + (end-start)/2
(4)       if (array[mid] =  element)  then return true
(5)       if (array[mid] > element) then
(6)            return(BinarySearch(array, start, mid-1, element))
(7)       else return(BinarySearch(array, mid+1, end, element))
```

   Argue that this algorithm is $O(lg(n))$.