**CS Comprehensive Exam**
**Fall 1998**
**Analysis of Algorithms**

This exam consists of **3 questions** that are worth 100 points. Please give **brief** and **precise** answers.

1. (**15 points**) True/False questions

   (a) A **Greedy** algorithm that solves a problem is usually **faster** than the **Dynamic** programming based algorithm that solves the same problem.

   (b) A **Greedy** algorithm that solves a problem is usually **more accurate** than the **Dynamic** programming based algorithm that solves the same problem.

   (c) Exhaustive search methods are usually the slowest out of all other problem solving methods.

   (d) In cryptology, **Private-key** based algorithms provide more security than **Public-key** based algorithms.

   (e) Factorizing large numbers into **primes** is very hard. Algorithms to do so could require more than a **century** of run time.

2. (**45 points**) Consider the following algorithm. (note: array indices start at 1)

```
procedure NEW-SORT (A[ ], n);
  begin
    if n > 1 do
      FIND-MAX(A[ ], n , max);
      swap(A[n], A[max]);
      NEW-SORT(A[ ], n − 1);
  end

procedure FIND-MAX (A[ ], n, index);
  begin
    index := 1;
    for i := 2 to n do
      if (A[i] > A[index]) index := i;
  end
```

(a) Is NEW-SORT a valid (correct) sorting algorithm?

(b) Is NEW-SORT stable?

(c) Calculate the time complexity of FIND-MAX.

(d) Give a recurrence that describes the time complexity of NEW-SORT.

(e) Solve the above recurrence.

(f) Which of the following sorting algorithms is closest to NEW-SORT in terms of processing data: Bubble, Insertion, Selection, Merge, Quick, Heap, Shell, Counting, Radix, Bin?

3. (**40 points**) Consider the following algorithm where a division by 3 is rounded down to the nearest integer (e.g., 8/3=7/3=6/3=2) and the indices of an array start at **zero**

```
procedure Foo (x, k[ ], lo, hi)
  begin
    if (lo > hi) then
      return (fail)
    else
      third = (lo + hi) / 3
      if (x == k[third]) then
        return (third)
      elseif (x < k[third]) then
        Foo(x, k, lo, third - 1)
      else
        Foo(x, k, third + 1, hi)
      endif
    endif
  end
```

(a) What is the value of $y$ after executing the statement

$$y := Foo(10, [3, 4, 9, 10, 12, 18, 20], 0, 6)$$

(b) What problem the above algorithm solves?

(c) What is the **best case** time complexity of Foo? Under what condition the best case occurs?

(d) What is the **worst case** time complexity of Foo? Under what condition the worst case occurs?

(e) Write a recurrence formula that represents the average case time complexity of Foo. (you do not need to solve it)