Exam 1
Computer Engineering 251
Santa Clara University
Thursday, 23 April 1998

NAME (Printed) ______________________________
NAME (Signed) ______________________________
Login ______________________________
Day of Lab Session ______________________________

Please show all your work and circle your answers. Your grade will be based on the the work shown.

Question 1	
Question 2	
Question 3	
Question 4	
Question 5	
Question 6	
Question 7	
Question 8	
Question 9	
Question 10	
TOTAL	

Useful Formulas

$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$

$\sum_{i=1}^{n} x^i = \frac{x^{n+1}-1}{x-1}$

$\sum_{i=1}^{n} \frac{1}{i} = \ln n + O(1)$

$\sum_{i=0}^{\infty} ix^i = \frac{x}{(1-x)^2}$

$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots$

$\lim_{n\to\infty}(1 + \frac{x}{n})^n = e^x$

$n! = \sqrt{2\pi n}(\frac{n}{e})^n(1 + \Theta(\frac{1}{n}))$

$n! = o(n^n)$

$n! = \omega(2^n)$

$\lg(n!) = \Theta(n \lg n)$

$\binom{n}{k} = \frac{n!}{k!(n-k)!}$

$(\frac{n}{k})^k \le \binom{n}{k} \le (\frac{en}{k})^k$

$E[X] = \sum_x xPr[X = x]$

$$Var[X] = E[(X - E[X])^2] = E[X^2] - E^2[X]$$

1. True or False (2 point each):

   (a) ___ $f(n) = O(g(n))$ implies $f(n) = o(g(n))$.

   (b) ___ $f(n) = o(g(n))$ implies $f(n) = O(g(n))$.

   (c) ___ $f(n) = O(g(n))$ implies $f(n) = \Theta(g(n))$.

   (d) ___ $f(n) = \Theta(g(n))$ implies $f(n) = O(g(n))$.

   (e) ___ $f(n) = \Theta(g(n))$ implies $f(n) = o(g(n))$.

   (f) ___ $f(n) = \omega(g(n))$ implies $f(n) = \Omega(g(n))$.

   (g) ___ $\lg(n!) = \Theta(\lg(n^n))$.

   (h) ___ $n! = \Theta(n^n)$.

   (i) ___ $2^n = \omega(n^2)$.

   (j) ___ $\lg n = \omega(\lg^* n)$.

   (k) ___ $n = o(n^2)$.

   (l) ___ $n = O(n^2)$.

   (m) ___ $3n^2 = \omega(n^2)$.

   (n) ___ $\lg n^2 = \Theta(\lg n)$.

2. Assume that every statement takes a constant c time. Give tight bounds on the order of growth and justify your answer:

   (a) Assume L is a singly linked list and k is a key:

   ```
   LIST-SEARCH(L,k)
   x <- head[L]
   while x != NIL and key[x] != k
        do x <- next[x]
   return x
   ```

   (b) Assume A is an array storing a heap and k is a key:

   ```
   HEAP-INSERT(A,k)
   heap-size[A] <- heap-size[A] + 1
   i <- heap-size[A]
   while i > 1 and A[PARENT(i)] < key
        do A[i] <- A[PARENT(i)]
           i <- PARENT(i)
   A[i] <- key
   ```

   (c) Assume A is an array:

   ```
   FIND-MAX(A)
   max <- - infinity
   for i <- 1 to n
        do if A[i] > max
           then max <- A[i]
   return max
   ```

3. Short Answer (2 points each):

   (a) What is the height of the binary search tree built from inserting keys from the sequence: $\{10, 3, 1, 12, 20, 18, 14, 16\}$?

   ____

   (b) What is the height of the heap built from inserting keys from the sequence: $\{10, 3, 1, 12, 20, 18, 14, 16\}$?

   ____

   (c) Suppose we use a random hash function h to hash n distinct keys into an array T of length n. What is the expected number of collisions? More precisely, what is the expected cardinality of $\{(x, y) \mid h(x) = h(y)\}$.

   ____

   (d) Suppose that we have numbers between 1 and 1000 in a binary search tree and want to search for the number 363. Could the following sequence be the sequence of nodes examined?

      i. 2, 252, 401, 398, 330, 344, 397, 363
      ii. 924, 220, 911, 244, 898, 258, 362, 363
      iii. 925,202, 911, 240, 912, 245, 363
      iv. 2, 399, 387, 219, 266, 382, 381, 278, 363
      v. 935, 278, 347, 621, 299, 392, 358, 363

   ____

   (e) To build a heap, instead of using a binary tree, you use a tree in which every node has up to 10 children. What is the height of the tree that stores a heap of n numbers?

   ____

4. Using the heap structure we discussed in class, write a function GET-SECOND-LEAST that takes as an argument the array that store the heap and returns the object with the second least key in the heap.

5. Using the heap structure we discussed in class, write a function GET-THIRD-LEAST that takes as an argument the array that store the heap and returns the object with the third least key in the heap.

6. Suppose that we have an array of n objects to sort and that the key of each record has the value 0 or 1. Give a simple, linear-time algorithm for sorting the n-objects in place. Use no storage of more than constant size in addition to the storage provided by the array.

7. Give asymptotic upper and lower bounds for $T(n)$ for the following two recurrences. Make your bounds as tight as possible, and justify your answers: Assume that $T(n)$ is constant for $n \leq 2$:

   (a) $T(n) = T(n/3) + n$

   (b) $T(n) = 4T(n/2) + n^2$

   (c) $T(n) = 4T(n/2) + n$

   (d) $T(n) = 5T(n/3) + 1$

   (e) $T(n) = 10T(n-2) + n$

   (f) $T(n) = T(\sqrt{n}) + 1$

   (g) $T(n) = T(n-1) + 1/n$

   (h) $T(n) = T(n-1) + \lg n$

8. In how many ways can n distinct objects be stored in circular list? Consider two lists the same if one can be rotated to form the other.
   Assume that the circular list has been implemented using an array with n elements and that the list is allowed to "wrap around" the end of the array.

9. In how many ways can 2^h distinct objects be stored in **heap** of height h? Consider two heaps the same if one can be obtained from the other by exchanging left and right subtrees of any node.

10. Suppose that you toss balls into b bins. Each toss is independent, and each balls is equally likely to end up in any bin. What is the expected number of ball tosses before at least one of the bins contains two balls?

11. Suppose that n balls are tossed into n bins, where each toss is independent and the ball is equlaly likely to end up in any bin. What is the expected number of empty bins? What is the expected number of bins with exactly one ball?

12. Assume $A[1..n]$ is an array and the following function is used to find the element with the maximum value in the array:

```
FIND-MAX(A)
max <- -infinity
for i <- 1 to n
     do if A[i] > max
          then max <- A[i]
return max
```

   (a) What are tight bounds on the **worst case** order of growth? Justify your answer:

   (b) What are tight bounds on the **best case** order of growth? Justify your answer:

   (c) What are tight bounds on the **average case** order of growth, assuming that all numbers in A are randomly drawn from the interval $[0, 1]$? Justify your answer:

13. (a) Design an algorithm that takes 3 ordered lists each of length $n/3$ and returns a merged list of these 3 lists.

    (b) Assume that each line of your algorithm above takes constant c time. Give tight upper bounds on the worst case running time of your algorithm. Justify your answer.

14. (a) Design an algorithm that partitions a list of size n into 3 sublists each of length $n/3$. The sublists have the property that all the elements in the first list are smaller than those in the second list, and all those elements in the second list are smaller than those in the third list. Your algorithm should set the values of $q1$ and $q2$ to be the indices of the the last value in the first and second lists, respectively. The prototype for the function is:

    ```
    THREE_PARTITION(A,p,q1,q2,r)
    ```

    where A is the array of objects, p and q are the starting and ending indices, and r1 and r2 are described above.

    (b) Assume that each line of your algorithm above takes constant c time. Give tight upper bounds on the worst case running time of your algorithm. Justify your answer.

15. Show that the running time of quicksort is $\Theta(n \lg n)$ when all elements of the array A have the same value.

16. Show that the running time of quicksort is $\Theta(n^2)$ when the array A is sorted in nonincreasing order.

17. Consider inserting the keys 10, 22, 31, 4, 15, 28, 17, 88, 59 into a hash table of length $m = 11$.

    (a) Using opening addressing with the primary hash function $h'(k) = k \text{mod} m$, illustrate the result of inserting these keys using linear probing:

    (b) Illustrate the result of inserting these same keys using double probing:

    $$h(k, i) = h'(k) + ih_2(i) \text{mod} m$$

    where h' is defined above and $h_2(k) = 1 + (k \text{mod} (m - 1))$.

18. Demonstrate the insertion of the keys 5, 28, 19, 15, 20, 33, 12, 17, 10 into a hash table. Assume the table has 9 slots and let the hash function be $h(k) = k \text{mod} 9$

    (a) First, draw the result of the insertions when collisions are resolved by chaining:

    (b) Illustrate the result of inserting these same keys if linear probing is used.