I certify that the work submitted with this exam is mine and was generated in a manner consistent with this document, the course academic policy on the course website, and the Rutgers academic code. I also certify that I will not disclose this exam to others who are taking this course and who will take this course in the future without the authorization of the instructor.

Date: _____

Name (please print): _____

Signature: _____

Name: _____ Section: _____

Problem Number(s)	Possible Points	Earned Points
1	10	
2	10	
3	10	
4	10	
5	10	
6	15	
7	5	
8	10	
9	10	
10	10	
Total	100	

Exam Time: 12 hours, 10 problems (14 pages, including this page)

- Write your name on this page and the last page, put your initials on the rest of the pages.
- If needed, use the last page to write your answer.
- Show your work to get partial credits.
- Show your rational if asked. Just giving an answer can't give you full credits.
- You may use any algorithms (procedures) that we learned in the class.
- Keep the answers as brief and clear as possible.

Name: _____ **Section:** _____

[Note: In this exam, both lg(x) and log(x) means $\log_2(x)$]
[Math facts: $\log a^b = b \log a$, $a^{\log b} = b^{\log a}$]

1. **(10 points) Asymptotic Growth of Functions**

List the 5 functions below in non-decreasing asymptotic order of growth:
$(\log n)^2$ $\log\log n$ n $n\log n$ $\log n$

(1) _____ (2) _____ (3) _____ (4) _____ (5) _____
smallest largest

2. **(10 points) Properties of O, Ω, Θ**

Clues:
(1) $f_1(n) = \Omega(n^2)$ (2) $f_2(n) = O(\log n)$ (3) $f_3(n) = \Theta(n)$
(4) $f_4(n) = O(2^n)$ (5) $f_5(n) = \Omega(n!)$

Circle TRUE (the statement must be always TRUE based on the clues above) or circle FALSE otherwise.

(a) $f_1(n) = \Omega(f_2(n))$ TRUE FALSE

(b) $f_1(n) = O(f_5(n))$ TRUE FALSE

(c) $f_2(n) = O(f_3(n))$ TRUE FALSE

(d) $f_3(n) = O(f_4(n))$ TRUE FALSE

(e) $f_5(n) = \Omega(f_4(n))$ TRUE FALSE

Name: _____ Section: _____

3. **(10 points) Recursion Trees**

Use the recursion tree method to determine the asymptotic upper bound of T(n). T(n) satisfies the recurrence T(n) = 2T(n-1) + n, and T(0)=0.

Name: _____ Section: _____

4. **(10 points) Solving Recurrences**
   (1) (5 points) Find a tight bound solution for the following recurrence:

$$T(n) = 4\,T(\tfrac{n}{2}) + c\,n \qquad (c \text{ is a positive constant})$$

   That is, find a function g(n) such that $T(n) \in \Theta(g(n))$. For convenience, you may assume that n is a power of 2, i.e., $n=2^k$ for some positive integer k. Justify your answer. [Note: Read question 4-(2) first before writing your answer]

Name: _____ Section: _____

(2) (5 points) Prove your answer in 4-(1) either using the iteration method or using the substitution method that we learned in our class.

Name: _____ Section: _____

5. **(10 points) Complexity of Sorting Algorithms**
   Sort a number of *n* integers. The value of the integers ranges from 0 to n^4-1. Please provide the worst-case running time for the following sorting algorithms (1) ~(5) provided in our lecture, using big-O or big-Θ notation wherever appropriate.

   (1) (1 point) Insertion Sort:

   (2) (1 point) Merge Sort:

   (3) (1 point) Quick Sort:

   (4) (1 point) Counting Sort:

   (5) (1 point) Radix Sort:

   (6) (5 points) What is the average-case running time for Merge Sort and Quick Sort, respectively? Why do we prefer Quick Sort algorithm in practice? Please provide at least two benefits of Quick Sort and also provide explanations for each benefit (plain language explanation is fine).

Name: _____ Section: _____

6. **(15 points) Stability of Sorting Algorithms**
   (1) (5 points) If a sorting algorithm is *stable*, what does it mean? Explain it clearly.

   (2) (5 points) Consider sorting an unsorted array A using the algorithms provided in the lecture, are merge sort and quick sort stable? Circle your answer below

Insertion Sort	Stable	Unstable
Merge Sort	Stable	Unstable
Quick Sort	Stable	Unstable
Counting Sort	Stable	Unstable
Radix Sort	Stable	Unstable

   (3) (5 points) In Radix Sort, why do we need to use a stable sorting algorithm to sort the numbers in each bucket? Provide a clear explanation.

Name: _____ Section: _____

7. **(5 points) Quick Sort and Algorithm Analysis**
   We have learned in class that while the expected running time of the randomized version of quicksort is O($n$log$n$), the worst-case running time is O(n^2). Show how quicksort can be made to run in O($n$log$n$) time in the worst case. Assume the input array is A[0:n-1] and all elements in A are distinct. Write your answer as pseudo-code and use plain language to explain the idea of your algorithm.
   (Hint: you can use any algorithm we have learned in the class as helper functions in parts of your designed algorithm. If you use an algorithm we learned in class as a helper function, you can directly call the function name in your pseudo-code without expanding the details of the helper function, as long as you clearly explain the meaning of the helper function.)

Name: _____ Section: _____

8. **(10 points) Randomized Algorithms and Hashing**

(1) (5 points) Show the result when we insert the keys 5; 28; 19; 15; 20; 33; 12; 17; 10 into a hash table with collisions resolved by linked list at each slot. Let the hash table have 9 slots, and let the hash function be $h(x) = (2x+1) \bmod 9$. (You are expected to draw the final hash table)

(2) (5 points) Suppose we hash elements of a set U of keys into m slots. Show that if $|U| > (n-1)m$, then there are at least n keys that all hash to the same slot, so that the worst-case searching time for hashing with linked list to resolve collisions is $\Theta(n)$.

Name: _____ **Section:** _____

9. **(10 points) Probability of Collision**
A frequently used hash family is the matrix multiplication hash family that we have introduced in class.

Suppose we have *n* buckets {*1, 2, …, n*}, we will use a binary string of length $b = \log_2 n$ to index each bucket. (For example, if we have 4 buckets, they will be indexed as 00, 01, 10, 11.)

Now, we would like to hash a u-bit binary string *x* into the hash table (For example, *x* could be an 8-bit string 10100110). To hash this u-bit string *x* into a bucket, we use the hash function $h_A(x) = (Ax) mod\ 2$, where *A* is a $b \times u$ dimensional binary matrix, and *x* is a $u \times 1$ dimensional column vector. As a result, $h_A(x)$ will be a $b \times 1$ dimensional vector which shows the bucket index that *x* will be hashed to. (For example, *x* is an 8-bit string $[10100110]^T$, and *A* is a 2×8 dimensional matrix $\begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$, then $h_A(x) = (Ax) mod\ 2 = [1, 0]^T$, which means that *x* will be hashed to bucket 2.)

Prove that the above hash function $h_A(x) = (Ax) mod\ 2$ is a <u>universal hash family</u>. [Hint: Consider hashing two arbitrary keys *x* and *y* into the hash table, both *x* and *y* are *u*-bit binary strings.]

Name: _____ Section: _____

Name: _____ Section: _____

10. **(10 points) Design an Algorithm**
You are given an array *A,* which stores *n* non-negative integers. Design an *efficient* **divide-and-conquer** algorithm that accepts *A* and *n* as inputs and returns the **index of the maximum value** in the array *A*.

(1) (5 points) **Basic idea and Pseudocode**: *(Pleas first use a few sentences of plan language to explain the basic idea of your algorithm, then provide the pseudo-code of your algorithm. Please be clear about each step of your pseudo-code, when necessary, you can add comments to explain each step)*

Name: _____ Section: _____

(2) (5 points) **Analysis:** derive a recurrence for the running time T(n) of your algorithm, and solve the recurrence to provide the asymptotic running time of your algorithm.

Name: _____ Section: _____

(You may use this page to write answers if needed. Please mark the problem number clearly)