This exam is closed book, notes, computer and cell phone.
Show partial solutions to get partial credit.
Make sure you have answered all parts of a question.
If your solutions are not written legibly, you won’t get full credit.
Clarity and succinctnes will be rewarded.

Question 1: __________ (out of 10)
Question 2: __________ (out of 10)
Question 3: __________ (out of 10)
Question 4: __________ (out of 10)
Question 5: __________ (out of 10)
Question 6: __________ (out of 10)
Question 7: __________ (out of 10)
Question 8: __________ (out of 10)
Question 9: __________ (out of 10)
Question 10: __________ (out of 10)

Total: __________ (out of 100)
1. Short questions:

(a) What is the running time of RadixSort as a function of
- the number of keys \( n \),
- the number of digits \( d \) in each key and
- the number of possible values \( k \) of a digit?

(b) What is recurrence for the worst-case behavior of QuickSort
and what is the worst-case running time?

(c) Somebody gives you \( n \) keys and you are to insert them into a Binary Search Tree.
How should you preprocess the keys and why?

(d) What is the definition of \( O \)-notation?

\[ f(n) = O(g(n)) \text{ if there are constants } c_1 \text{ and } c_2 \text{ s.t. } \ldots \]
2. Somebody proposes a sorting algorithm using Red Black Trees:

   **Insert the \( n \) keys and then print out the keys in inorder.**

   Is this a good sorting algorithm?
   Discuss its advantages and disadvantages.
   In particular, discuss
   - its worst-case running time,
   - average case running time, and
   - space requirement
   in relation to HeapSort and QuickSort.
3. Show by induction (on the height of the tree) that the number of nodes in a complete ternary tree of height $h$ is exactly

$$3^{h+1} - 1$$

$$2$$.

(In such a tree all internal nodes have exactly 3 children.)

Hint: Begin by drawing the complete ternary trees of height 0, 1, 2.
4. A simple hash function for collision resolution by chaining is

\[ h(k) = k \mod m, \]

where \( k \) is the key and \( m \) the number of slots/chains.

Give some properties of a good hash function and present a alternate hash function that could be used.

What are convenient choices for \( m \) for your alternate hash function?
5. What operation are efficiently implementable on a Binomial Heap? Give a list of operations and the time bound in big O notation.

Show how to merge the following two BH’s A and B. Hint: What binary number do A and B correspond to?

Figure 1: Two binomial heaps, A and B.
6. Show how to sort $n$ integers in the range from 0 to $n^2 - 1$ in $O(n)$ time using RadixSort.
7. What is binary-search-tree property?

What is the min-heap property?

What is the time to build a BST with \( n \) arbitrary keys (worst-case and average case)?

What is the time to build a min-heap with \( n \) arbitrary keys?

Can the min-heap property be used to print the keys in the tree in sorted order in \( O(n) \) time? Show how or explain the contradiction to a known theorem about sorting.
8. Show that

$$\sum_{i=1}^{n} \log_2(4i) = \Theta(n \log(n))$$.

We suggest to reason the upper and lower bound separately.
9. Show the red-black trees that result after successively inserting the keys

\[ 41 \ 38 \ 31 \ 12 \ 19 \ 8 \]

into an initially empty red-black tree.
10. Solve the following recurrences using the Master’s Theorem. In each case, reason which case of the Master theorem (given below) applies and why and determine the solution in $\Theta$ notation.

$$T(n) = 2T\left(\frac{n}{2}\right) + n$$ \hspace{1cm} (1)

$$T(n) = 3T\left(\frac{n}{4}\right) + n^2$$ \hspace{1cm} (2)

$$T(n) = 2T\left(\frac{n}{4}\right) + n$$ \hspace{1cm} (3)

Master’s Theorem

Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the recurrence

$$T(n) = aT\left(\frac{n}{b}\right) + f(n).$$

Then $T(n)$ has the following asymptotic bounds:

(a) If $f(n) = O(n^{\log_b(a)-\varepsilon})$ for some constant $\varepsilon > 0$, then $T(n) = \Theta(n^{\log_b(a)})$.

(b) If $f(n) = O(n^{\log_b(a)})$, then $T(n) = \Theta(n^{\log_b(a)} \log(n))$.

(c) If $f(n) = \Omega(n^{\log_b(a)+\varepsilon})$ for some constant $\varepsilon > 0$,

and if $af\left(\frac{n}{b}\right) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large $n$, then $T(n) = \Theta(f(n))$. 
