Final Exam

- Do not open this quiz booklet until directed to do so. Read all the instructions on this page.
- When the quiz begins, write your name on every page of this quiz booklet.
- You have **180 minutes to earn 180 points.** Do not spend too much time on any one problem. Read them all first, and attack them in the order that allows you to make the most progress.
- **You are allowed a 3-page cheat sheet**. No calculators or programmable devices are permitted. No cell phones or other communications devices are permitted.
- Write your solutions in the space provided. If you need more space, use **the back of the sheet** containing the problem. Pages may be separated for grading.
- Do not waste time and paper rederiving facts that we have studied. Simply cite them.
- When writing an algorithm, a **clear** description in English will suffice. Pseudo-code is not required unless you find that it helps with the clarity of your presentation.
- **Pay close attention to the instructions for each problem**. Depending on the problem, partial credit may be awarded for incomplete answers.

Problem	Parts	Points	Grade	Grader
0	2	2		
1	9	18		
2	3	20		
3	2	15		
4	3	15		
5	1	20		
6	1	20		
7	1	20		
8	2	20		
9	2	30		
Total		180		

Name: ____________________

Circle your recitation:	R01 Daniel Manesh **10AM**	R02 Jennifer Jang **10AM**	R03,6 Adam Yedidia **11,12PM**	R04,5 Kevin Chen **11,12PM**	R07 Casey O'Brien **1PM**	R08 Jonathan Chien **2PM**	R09 Victor Valov **3PM**

Problem 0. What is Your Name? [2 points] (2 parts)

(a) [1 point] Flip back to the cover page. Write your name there.

(b) [1 point] Flip back to the cover page. Circle your recitation section.

Problem 1. True or False [18 points]

For each of the following questions, circle either T (True) or F (False). There is no need to justify the answers.

Each correct answer is worth 2 points and each incorrect answer receives -2 points. If the total is negative, you will receive 0 points for this problem.

(a) T F Suppose algorithm FAST runs in $\Theta(n^2)$ and algorithm SLOW runs in $\Theta(2^n)$. FAST will run faster than SLOW on any input.

(b) T F It is possible to merge k sorted lists of objects, each of size n, into a single sorted list of size nk in $O(nk)$ time using subroutines that compare a given pair and copy a given object in one time step.

(c) T F A balanced binary search tree can be augmented to implement all priority queue operations (MIN, EXTRACT-MIN, INCREASE-KEY, INSERT) at least as fast (asymptotically) as a min-heap.

(d) T F For all positive integers a, b, and m, we have $(a \bmod m)(b \bmod m) = ab \bmod m$.

(e) T F For all odd primes p such that $\gcd(3, p-1) = 1$, and for all $a \in \mathbb{Z}_p^*$, if x is the inverse of 3 modulo $p-1$, then a^x is a cube root of a modulo p.

(f) T F For all primes p that end in 99 (such as 199), if a is a square modulo p, then $\sqrt{a} = a^{(p+1)/2} \pmod{p}$.

(g) T F For a directed graph, the absence of back edges with respect to a DFS tree implies that the graph is acyclic.

(h) T F At least one of the following statements is in NP: 1) Given undirected graphs G and H, the graphs are isomorphic; 2) Given undirected graphs G and H, the graphs are not isomorphic.

(i) T F There is a polynomial time verifiable NP proof for the following statement: "Given positive integers n and k such that $1 < k < n$, there is no integer m such that $1 < m \leq k$ and n is divisible by m."

Problem 2. Short answers [20 points]

(a) [5 points] You are given a weighted, directed graph $G = (V, E)$ with n vertices and positive integer edge weights. Given a node t and an integer W, design an algorithm running in time $O(n^2 \log n)$ which determines, for every $s \in V$, if there exists a path of weight less than W from s to t.

(b) [5 points] Give an algorithm with strictly faster asymptotic running time than the trivial $\Theta(n)$ for the following problem: Given an array of n integers $A = [a_1, a_2, \ldots, a_n]$ such that the number of inversions, defined as pairs of indices (i, j) where $i < j$ but $a_i > a_j$, is at most $\log_2 n$, and given an integer x, decide whether x is an element of A.

(c) [10 points] Given an undirected weighted graph $G = (V, E)$ with non-negative weights, and two vertices $s, t \in V$, design an efficient algorithm that returns the shortest-weight path from s to t among all possible paths that take no more than 5 edges; or returns `None` if no such path exists. Full credit will be given to algorithms that run in time $O(n^2 \log n)$, where n is the number of vertices.

Problem 3. Dynamic programming [15 points]

You are given a **rooted tree** $G = (V, E)$, where for each vertex v, there is an associated positive integer reward r_v. The goal is to compute $S \subseteq V$ which maximizes the total reward, but which also does not contain any adjacent nodes. That is, we want to choose $S = \{v_1, v_2, \dots, v_k\}$ to maximize $\sum_{v \in S} r_v$ with the constraint that if (u, v) is an edge in our graph, then S can't contain both u and v.

You decide to solve this problem using Dynamic Programming, using the following subproblems:

$R[v, True] :=$ The maximum total reward from the subtree rooted at v, where v is included in S.

$R[v, False] :=$ The maximum total reward from the subtree rooted at v, where v is **not** included in S.

(a) [5 points] Write down the values for $R[v, True]$ and $R[v, False]$ when v is a leaf.

$R[v, True] =$

$R[v, False] =$

(b) [10 points] Write down the recurrence for $R[v, True]$ and $R[v, False]$ when v is **not** a leaf.

$R[v, True] =$

$R[v, False] =$

Problem 4. Hashing [15 points]

Ben Bitdiddle is investigating the performance of different hash table collision strategies. His universe of possible inputs is $U = \{0, 4, 8, 12, 16, ..., 100\}$. His hash function is

$$h(k) = k \mod m,$$

where m is the size of the table.

(a) [5 points] Ben inserts the specific elements 32, 12, 4, and 76 into the table (in the specified order). Indicate the contents of each entry in a hash table of size 7 after inserting the above elements with chaining as the collision resolution strategy.

Hash Entry	Key(s)
0	
1	
2	
3	
4	
5	
6	

(b) [5 points] Repeat part **(a)**, but this time use linear probing as the collision resolution strategy.

Hash Entry	Key(s)
0	
1	
2	
3	
4	
5	
6	

(c) [5 points] Ben notices the load factor of the table is large and proposes resizing the table to size 20. Is this a good choice of table size considering the elements that may be inserted? Explain why or why not.

Problem 5. Hanging Posters [20 points]

A company is hanging advertisement posters in a long hallway. There are n possible distinct locations $x_1, \ldots, x_n$ for hanging posters where each x_i is an integer that specifies the distance of the ith location (in meters) from the entrance of the hallway. For each $i \in \{1, \ldots, n\}$, hanging a poster at location i brings a revenue of r_i for the company, where r_i is a positive integer. In order to maximize the revenue, the company desires to hang as many posters as possible, however, there is the limiting constraint that no two posters can be within five meters of each other.

Design an efficient algorithm that takes the locations $x_1, \ldots, x_n$ (in no particular order) and returns the maximum total revenue that can be obtained from choosing any subset of the locations. Full credit will be given to algorithms that run in $O(n \log n)$.

For example, suppose that $n = 4$, and $(x_1, x_2, x_3, x_4) = (6, 8, 14, 12)$, and $(r_1, r_2, r_3, r_4) = (5, 6, 1, 5)$. Then, the optimal solution is to place posters at x_1 and x_4 for a total revenue of $5 + 5 = 10$.

Problem 6. Palindromes [20 points]

Given a string $x = x_1, ..., x_n$, design an efficient algorithm to find the minimum number of characters that need to be inserted to make it a palindrome (recall that a palindrome is a string such as "racecar" the reads the same backwards). Analyze the running time of your algorithm and justify its correctness.

For example, when $x =$ "ab3bd", we need to insert two characters (one "d" and one "a"), to get either the of the palindromes "dab3bad" or "adb3bda".

Problem 7. Housekeeping [20 points]

Ben Bitdiddle likes to make a little extra money doing tasks around the house. Each task has an associated time t that it will take Ben to do the task, and a value v, representing the amount of money that Ben will be paid to do it.

Ben wants to design a data structure to maintain the set of tasks he can perform. Whenever someone thinks of a new task for Ben to do, Ben inserts the task into the structure. Whenever Ben has some amount of free time t, he wants to query the structure to determine the maximum value he can earn from a single task in time at most t. Assume that Ben never needs to remove a task from the structure.

Specifically, design a data structure which supports the following operations:

- INSERT(t, v): Insert a task into the structure which takes time t and has value v.
- QUERY(t): Return the maximum value that Ben can get from performing a single task in time at most t.

Both operations should run in $O(\log n)$, where n is the number of tasks in the structure at the time that the operation is called. For this problem, **no** partial credit will be awarded for slower solutions.

Problem 8. Congruence Systems [20 points]

(a) [10 points] Prove or disprove the following: There is a positive integer x satisfying both of the following equations.

$$x \equiv 1 \pmod{6}$$
$$x \equiv 2 \pmod{15}.$$

(b) [10 points] Given four positive integers a_1, a_2, b_1, and b_2, all of which are no more than 2^n, give an algorithm running in polynomial time in n that returns `True` if there exists some x satisfying:

$$x \equiv a_1 \pmod{b_1}$$
$$x \equiv a_2 \pmod{b_2}$$

and `False` if no such x exists. Note that b_1 and b_2 need are not necessarily be relatively prime.

Problem 9. Sensible Subsets [30 points]

A set of integers S is called *k-sensible* if the sum of any k numbers in S is greater than or equal to every number in S.

(a) [10 points] Given an array A of n distinct positive integers, prove that the maximum size *k-sensible* subset S of A contains consecutively valued numbers of A. In other words, if S contains a and b and $a < b$, then S must also contain every number x in A such that $a < x < b$. Assume that the maximum size *k-sensible* subset S of A is unique.

(b) [20 points] Give an algorithm that takes an array A and integer k and returns the maximum size k-sensible subset of A. Analyze your runtime in terms of n (you may assume $k \leq n$). Full credit will be given to algorithms that run in $O(n \log n)$.

SCRATCH PAPER

SCRATCH PAPER