Final

- 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 the top of every page of this quiz booklet.
- You have 180 minutes to earn a maximum of 180 points. Do not spend too much time on any one problem. **Read through all problems first**, then solve them in an order that allows you to make the most progress.
- **You are allowed three double-sided letter-sized sheet with your own notes**. No calculators, cell phones, or other programmable or communication devices are permitted.
- Write your solutions in the space provided. Pages will be scanned and separated for grading. If you need more space, write "Continued on S1" (or S2, S3, S4, S5, S6, S7) and continue your solution on the referenced scratch page at the end of the exam.
- Do not waste time and paper rederiving facts that we have studied in lecture, recitation, or problem sets. Simply cite them.
- When writing an algorithm, a **clear** description in English will suffice. Pseudo-code is not required. Be sure to **briefly** argue that your **algorithm is correct**, and analyze the **asymptotic running time of your algorithm**. Even if your algorithm does not meet a requested bound, you **may** receive partial credit for inefficient solutions that are correct.

Problem	Parts	Points
0: Information	2	2
1: Decision Problems	10	40
2: Network Upgrades	2	20
3: Sort by Number	2	18
4: Isomorphic Sentences	1	15
5: Sidestepping Smells	1	15
6: MonoMono Manors	1	15
7: Plant Peeping	1	20
8: JobBunny	1	15
9: DJ Ikstra Playlist	1	20
Total		180

Name: _____

School Email: _____

Problem 0. [2 points] **Information** (2 parts)

(a) [1 point] Write your name and email address on the cover page.

(b) [1 point] Write your name at the top of each page.

Problem 1. [40 points] **Decision Problems** (10 parts)

For each of the following questions, circle either **T** (True) or **F** (False), and **briefly** justify your answer in the box provided (a single sentence or picture should be sufficient). Each problem is worth 4 points: 2 points for your answer and 2 points for your justification. **If you leave both answer and justification blank, you will receive 1 point**.

(a) T F Selection sort always performs at least as many comparisons as insertion sort.

(b) T F If $T(n) = 4T(n/2) + O(n^2 \log^2 n)$ and $T(1) = O(1)$, then $T(n) = O(n^2 \log^2 n)$.

(c) T F A max heap can be converted into a min heap in linear time.

(d) T F Performing a single rotation on a binary search tree always results in binary tree that also satisfies the BST Property.

(e) T F If a node a in an AVL tree is not a leaf, then a's successor is a leaf.

(f) T F When storing n integer keys in a hash table, using a hash function randomly chosen from a universal hash family, there will be no collisions in the hash table if all keys are distinct.

(g) T F Johnson's Algorithm relaxes each edge in a graph $G = (V, E)$ at most $|V|$ times.

(h) T F Let dynamic programming subproblem $x(i)$ depend on subproblems $x(1), x(2), \ldots, x(i-1)$. In a top-down implementation, the value $x(i)$ is computed before $x(i-1)$, and in a bottom-up implementation, $x(i-1)$ is computed before $x(i)$.

(i) T F Suppose an algorithm runs in $\Theta(ns)$ time for any input array A of n positive integers where $\max(A) = s$. Then the algorithm runs in polynomial time.

(j) T F If some NP-complete problem is EXP-hard, then P $\neq$ NP.

Problem 2. [20 points] **Network Upgrades**

Acrosoft Mizure is a cloud computing company that maintains a network of n computers. Mizure has connected p pairs of computers using wired links: a **wired link** is a direct two-way wired connection between two computers. The **wired connectivity** of a computer is the number of computers it can talk to via any sequence of wired links.

(a) [10 points] Describe an efficient[1] algorithm to find a largest subset of **existing** wired links that could be **removed** without changing any computer's wired connectivity.

(b) [10 points] Describe an efficient[1] algorithm to find a smallest set of **new** wired links that, if **added** to the network, would maximize each computer's wired connectivity.

[1] By "efficient", we mean that faster correct algorithms will receive more points than slower ones.

Problem 3. [18 points] **Sort by Number**

In each part, describe an algorithm to sort n **pairs of integers** $(a_1, b_1), \ldots, (a_n, b_n)$ by increasing $f(a_i, b_i)$ value, for each function f described below. If your algorithms use division, they should only use **integer** division (Python's `//` and `%`).

(a) [8 points] If $f(a, b) = \left(1 + \frac{a}{n}\right) \cdot 2^b$, and all input pairs (a_i, b_i) satisfy $0 \leq a_i < n$ and $0 \leq b_i < n$, show how to sort pairs in $O(n)$ time.[2] Note that $f(a, b) < f(0, b + 1)$ for all valid a and b, because $1 + \frac{a}{n}$ is always less than 2.

(b) [10 points] If $f(a, b) = a/b$, and all input pairs (a_i, b_i) satisfy $0 \leq a_i < n$ and $0 < b_i < m < n$, then show how to sort pairs in $O(n \log m)$ time.

[2] This function essentially corresponds to how computers represent floating-point numbers.

Problem 4. [15 points] **Isomorphic Sentences**

In this problem, a **string** is a sequence of lowercase English letters, and a **sentence** is a sequence of strings. Assume that each string is storable in a **constant number** of machine words. Two sentences $A = (a_1, \ldots, a_k)$ and $B = (b_1, \ldots, b_k)$ are **isomorphic** if they have the same ordered pattern of string repetitions, i.e., $a_i = a_j$ if and only if $b_i = b_j$, for all i, j. For example, sentence A below is isomorphic to sentence B, but A is not isomorphic to sentence C.

$$\begin{aligned} A = (&\texttt{i}, &&\texttt{like}, &&\texttt{that}, &&\texttt{you}, &&\texttt{like}, &&\texttt{algorithms})\\ B = (&\texttt{my}, &&\texttt{dog}, &&\texttt{has}, &&\texttt{many}, &&\texttt{dog}, &&\texttt{friends})\\ C = (&\texttt{that}, &&\texttt{example}, &&\texttt{is}, &&\texttt{a}, &&\texttt{silly}, &&\texttt{example}) \end{aligned}$$

Given a list of n sentences, each containing exactly k strings, describe an $O(nk)$-time algorithm to partition them into groups such that two sentences are in the same group if and only if they are isomorphic. Specify whether your running time is expected, amortized, or worst-case.

Problem 5. [15 points] **Sidestepping Smells**

Cinnamon City consists of n locations where pairs of locations are connected by two-way roads, and each location is incident to at most 5 roads. Some locations contain a trash dumpster. Each dumpster d_i has a different **smell radius** r_i. Someone at a location can **smell** a dumpster d_i if and only if they can walk from their location at most r_i feet along roads to d_i. Orcas the Grouch wants to walk across town while avoiding the smell of trash from dumpsters. Given a map of Cinnamon City marked with the length of each road, and the location and smell radius of each dumpster, describe an efficient[3] algorithm to find the shortest route from 600 Rosemary Rd. to 6 Allspice Ave., along which Orcas will never smell trash, if such a route exists.

[3] By "efficient", we mean that faster correct algorithms will receive more points than slower ones.

Problem 6. [15 points] **MonoMono Manors**

The new **single-player** board game *MonoMono* consists of n **manors** $M = \{m_0, m_1, \ldots, m_{n-1}\}$ arranged in a directed cycle. The player begins at manor m_0 with d dollars (where d is a positive integer), and then repeatedly takes a turn. Each turn, the player moves between 2 and 12 spaces forward along the directed cycle, based on a roll of the dice. For example, if $n = 10$ and the first two rolls are 6 and 8, then the player will first move to m_6 and then move to m_4. The player owns a known subset $M_p \subset M$ of manors, and manor ownership never changes. Each manor m_i has an integer **transaction cost** c_i: when the player moves to manor m_i, the player gains c_i dollars if the player owns the property, and loses c_i dollars otherwise. Describe an $O(nd)$-time algorithm to determine the minimum number of turns the player could possibly take to at least **double their starting money, without ever going bankrupt** (having a negative balance).

Problem 7. [20 points] **Plant Peeping**

Nevergreen Terrace is a straight flat street that starts at the dangerous Fallfield Nuclear Power Plant located at address 0. Tall buildings are frequently built and demolished along Nevergreen Terrace, and a building's property value is higher when another building shields it from view of the plant. Specifically, a building b can **see** the power plant if b is **strictly taller** than every building strictly between b and the power plant. Describe a database to maintain the set of buildings along Nevergreen Terrace, supporting the following three operations each in worst-case $O(\log n)$ time, where n is the number of buildings on Nevergreen Terrace at the time of the operation.

- `construct(b)`: record a new building b with height $b.h$ (an integer number of meters) at address $b.a$ (an integer number of meters from the power plant).
- `demolish(a)`: remove the building at address a from the database.
- `can_see_plant(a)`: return `True` if the building at address a is **strictly taller** than every building having positive address less than a, and `False` otherwise.

Problem 8. [15 points] **JobBunny**

Rager Robbit is a freelancer who works independent jobs posted on the new job website, JobBunny. There are n jobs currently listed on the site. Each job has a start date, an end date, and the number of dollars that Rager would earn by working on that job every day from its start date up to and including its end date. Multiple jobs may have the same start or end date. Rager is bad at multitasking, so he cannot work multiple jobs on the same day. Describe an $O(n \log n)$-time **dynamic-programming** algorithm to determine which JobBunny jobs Rager should work on in order to maximize his earnings. Slower correct algorithms will receive partial credit.

Problem 9. [20 points] **DJ Ikstra Playlist**

DJ Ikstra is preparing a playlist for a music festival. She has a list of n songs in her library, and has tagged each song with its duration in integer seconds, its genre (either house, techno, or dubstep), and its **awesomeness**: a positive integer indicating how awesome she thinks it is (no two songs are equally awesome). The festival organizers have allocated only r seconds for her playlist, which will consist of a sequence of songs played back to back. DJ Ikstra wants her playlist to be both elevating and diverse. A playlist is **elevating** if the songs in playlist order never decrease in awesomeness; and is **diverse** if no two consecutive songs belong to the same genre. Describe an $O(n \log n + nr)$-time **dynamic-programming** algorithm to find an elevating and diverse playlist lasting at most r seconds that maximizes the total awesomeness of all songs played.

SCRATCH PAPER 1. DO NOT REMOVE FROM THE EXAM.

You can use this paper to write a longer solution if you run out of space, but be sure to write "Continued on S1" on the problem statement's page.

SCRATCH PAPER 2. DO NOT REMOVE FROM THE EXAM.

You can use this paper to write a longer solution if you run out of space, but be sure to write "Continued on S2" on the problem statement's page.

SCRATCH PAPER 3. DO NOT REMOVE FROM THE EXAM.

You can use this paper to write a longer solution if you run out of space, but be sure to write "Continued on S3" on the problem statement's page.

SCRATCH PAPER 4. DO NOT REMOVE FROM THE EXAM.

You can use this paper to write a longer solution if you run out of space, but be sure to write "Continued on S4" on the problem statement's page.

SCRATCH PAPER 5. DO NOT REMOVE FROM THE EXAM.

You can use this paper to write a longer solution if you run out of space, but be sure to write "Continued on S5" on the problem statement's page.

SCRATCH PAPER 6. DO NOT REMOVE FROM THE EXAM.

You can use this paper to write a longer solution if you run out of space, but be sure to write "Continued on S6" on the problem statement's page.

SCRATCH PAPER 7. DO NOT REMOVE FROM THE EXAM.

You can use this paper to write a longer solution if you run out of space, but be sure to write "Continued on S7" on the problem statement's page.