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. Skim them all first, and attack them in the 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 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.
- **Pay close attention to the instructions for each problem**. Depending on the problem, partial credit may be awarded for incomplete answers.

Problem	Parts	Points
0: Information	2	2
1: Decision Problems	10	40
2: Sorting Sorts	2	24
3: Pams vs. Ratriots	1	14
4: Costly Campaigning	1	16
5: Parkour Performance	1	16
6: Mostly-Blue	1	16
7: Zero Sum	2	32
8: Soldier Supply	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 If $f(n) \in O(3^{2n})$, then $f(n) \in \Omega(2^{3n})$.

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

(c) T F Given n positive integers, where each is at most u, Radix sort sorts them by breaking each integer into $O(\log_{10} u)$ base-10 digits, and then sorts them by their digits in least-significant order using counting sort.

(d) T F Inserting n^2 items sequentially to the back of an empty dynamic array takes worst-case $O(n^2)$ time.

(e) T F Given a Set AVL tree storing n keyed items ordered by key, one can construct a key-ordered max-heap on the same n items in worst-case $O(n)$ time.

(f) T F Given a weighted connected undirected graph $G = (V, E)$ containing exactly $|V| - 1$ edges, one can solve weighted Single-Source Shortest Paths from any $s \in V$ in $O(|V|)$ time.

(g) T F Given a weighted simple directed graph $G = (V, E)$, where every vertex connects to at least $|V|/2$ edges and every edge has strictly negative weight, one can find a maximum-weight directed path from vertex $s \in V$ to $t \in V$ in linear time.

(h) **T F** While running Bellman-Ford from vertex s in a simple graph, suppose distance estimate $d'[s, v]$ is assigned a new smaller value D in the k^{th} round of relaxation. Then D is the weight of a path from s to v traversing exactly k edges.

(i) **T F** If a decision problem A is NP-hard, then A is also NP-complete.

(j) **T F** If a decision problem A is NP-hard and $A \in \mathsf{P}$, then $\mathsf{P} = \mathsf{NP}$.

Problem 2. [24 points] **Sorting Sorts**

(a) [12 points] The ***x*-value** of a pair (a, b) is the real number $\sqrt{a}+b\sqrt{x}$. Let B be an array of n pairs of positive integers (a_i, b_i) with $a_i < n$ and $b_i < n^3$ for all $i \in \{1, \ldots, n\}$. Describe an $O(n)$-time algorithm to sort the pairs in B by their x-values for $x = n$.

(b) [12 points] Let A be an array $(a_1, \ldots, a_n)$ of n positive integers, where $\lfloor \lg n \rfloor$ adjacent pairs in A appear in sorted order[1]. Describe an $O(n \log \log n)$-time algorithm to sort A.

[1]i.e., $|\{a_i \mid i \in \{2, \ldots, n\} \text{ and } a_{i-1} < a_i\}| = \lfloor \lg n \rfloor$

Problem 3. [14 points] **Pams vs. Ratriots**

Bom Trady is a football fan who will be attending the Big Game$^{\text{TM}}$, where her home team, the Ratriots, will be playing rival team, the Pams. The game will be held in a neutral city, so Bom will fly there and walk to the stadium from the airport.

- Upon arriving, Bom discovers some streets are **occupied**. A street occupied by fans from team A may only be traversed while wearing a team A jersey, and possessing no other jersey.
- Bom has a map depicting the m two-way streets and n intersections in the city, where each street directly connects two intersections, and each intersection connects at most four streets. The airport and stadium intersections are both marked on the map.
- Each street is marked with its positive integer length and its occupied status: either Ratriots, Pams, or unoccupied. Some intersections on the map are marked as having shops, which sell jerseys from both teams.

Assume Bom begins wearing a Ratriots jersey, always has enough money to purchase jerseys, and may discard a jersey at any time. Given her map, describe an $O(n \log n)$-time algorithm to return a shortest occupation-respecting route from the airport to the stadium (or return no such route exists).

Problem 4. [16 points] **Costly Campaigning**

Welizabeth Arren is campaigning in a Massachusetts (MA) election in the year 2048. Due to rising sea-levels, Massachusetts has become a line of n towns $(t_1, \dots, t_n)$, bounded by water on either side. Arren wants to run TV ads in some of the towns: MA residents love to gossip, so if Arren runs an ad in the town named t_i, the k cities on either side[2] of t_i will also hear about Arren's campaign. Each town t_i has a known positive integer cost c_i to broadcast an ad. Describe an $O(nk)$-time algorithm to determine a subset of towns in which to run TV ads, so as to minimize total broadcast costs, while ensuring that every town in MA hears about Arren's exciting plan.

[2]Specifically, towns $(t_{\max(i-k,1)}, \dots, t_i, \dots, t_{\min(i+k,n)})$ will hear about Arren's campaign.

Problem 5. [16 points] **Parkour Performance**

Parkour athlete Diane Royal is planning a performance on top of an $n \times n$ square grid of platforms, where each platform p_{ij} has a known positive integer height $h(p_{ij})$ for $i, j \in \{1, \ldots, n\}$. Diane has a single ladder which she can place between any two adjacent[3] platforms at the start of the performance, and she may climb between them, up the ladder, **at most once** during the performance. Diane will start her performance on some platform, and then repeatedly either:

- **jump** from her current platform a to an adjacent platform b with strictly lower height[4], or
- **climb** the ladder she placed at the start, if her current platform is adjacent to the ladder.

Jumping from a platform a to a square b with $h(b) < h(a)$ will result in $(h(a) - h(b))^2$ **coolness**, but climbing the ladder yields no coolness. Given the height of each platform, describe an $O(n^2)$-time algorithm to determine: (1) where she should place her ladder and (2) where she should start, so as to maximize the total sum of coolness resulting from her performance.

[3] Two platforms p_{ij} and p_{xy} are adjacent if and only if $|i - x| + |j - y| = 1$.

[4] i.e., $h(b) < h(a)$

Problem 6. [16 points] **Mostly-Blue**

- A **blue-labeled** graph is a directed graph where every edge is labeled either blue or not-blue.
- A cycle in a blue-labeled graph is **mostly-blue** if more than half of its edges are blue.
- A vertex is **high-degree** if the sum of its in-degree and out-degree is strictly more than two.

Given a blue-labeled simple graph G containing n vertices and at most $O(n)$ edges, where exactly $k < n$ vertices are high-degree, describe an $O(n + k^3)$-time algorithm to determine whether G contains a mostly-blue cycle.

Problem 7. [32 points] **Zero Sum**

(a) [16 points] Given $2k$ arrays, where each array contains exactly n integers, describe an $O(n^k)$-time algorithm to determine whether there exists $2k$ integers, exactly one from each array, that sum to zero. State whether your running time is worst-case, expected, and/or amortized. Significant partial credit will be given for $O(kn^k)$-time algorithms. (Hint: First try solving for small k.)

(b) [16 points] Now suppose the absolute value of any integer in the $2k$ arrays is less than u. Describe an $O(nuk^2)$-time algorithm to determine whether there exists $2k$ integers, exactly one integer from each array, whose sum is zero.
(Hint: use dynamic programming!)

Problem 8. [20 points] **Soldier Supply**

Savos Deaworth is a general who commands n soldiers.

- Savos has assigned each soldier i a pair of positive integers: a **troop number** t_i and a unique **soldier ID** d_i, and has stored all n soldier pairs (t_i, d_i) in a length-n array S.
- At the start of battle, Deaworth sends soldiers to the front, to fight side-by-side in a line. The initial **front line** is provided in an array L: $|L| < n$ soldier IDs listed in a **specified** order.
- Sometimes, a soldier may be temporarily **wounded** and must be removed from the front.
- Other times, Savos may **send** a new soldier to the front line, and will need to tell them where to stand: specifically, how many soldiers should be on their left when they go to the front.
- A soldier is more effective if they fight **next to** another soldier having the same troop number[5].

Describe a database supporting the following **worst-case** operations:

`init(S, L)`	initialize with soldier array `S` and initial front line `L` in $O(n \log n)$ time
`wound(d)`	remove the soldier with ID `d` from the front line in $O(\log n)$ time (assume `d` is in the front line before the operation)
`send(d)`	send and place the soldier with ID `d` into the front line in $O(\log n)$ time place `d` next to a soldier in the front line having the same troop number if possible if no soldier in front line with the same troop number, place `d` in left-most position **return** the number of soldiers to the left of where `d` is placed (assume `d` is not in the front line before the operation)

[5]Two soldiers fight next to each other if they are adjacent in the front line.

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.