Quiz 3

- 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 60 minutes to earn a maximum of 60 points. Do not spend too much time on any one problem. Skim them all first, and work on 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) 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: Pseudo-Circling	3	3
2: Number Scrabble	1	17
3: Limited-Unlimited	1	19
4: Office Hour Optimization	1	19
Total		60

Name: ____________________

School Email: ____________________

Problem 1. [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.

Please solve problems (3), (4), and (5) using **dynamic programming**. Be sure to define a set of subproblems, relate the subproblems recursively, argue the relation is acyclic, provide base cases, construct a solution from the subproblems, and analyze running time. Correct but inefficient dynamic programs will be awarded significant partial credit.

Problem 2. [3 points] **Pseudo-Circling**

Indicate whether the given running times of each of problems (3), (4), and (5) are polynomial or pseudopolynomial by circling the appropriate word below. **One can answer this question without actually solving problems (3), (4), and (5).** (1 point each)

(a) Problem 3: Number Scrabble Polynomial Pseudopolynomial

(b) Problem 4: Limited-Unlimited Polynomial Pseudopolynomial

(c) Problem 5: Office Hour Optimization Polynomial Pseudopolynomial

Problem 3. [17 points] **Number Scrabble**

Number Scrabble is a one-player game played on an array $T = [t_0, \dots, t_{n-1}]$ of n positive integers. There is a list $P = \{(p_0, v(p_0)), \dots, (p_{m-1}, v(p_{m-1}))\}$ of m unique **playable words**, where playable word p_i is a non-empty array of at most 10 positive integers and $v(p_i)$ is the positive integer value of p_i. The objective of the game is to find a **gameplay** S — a list of **non-overlapping** subarrays (i.e., substrings) of T, each a playable word — where S has maximum total value, $\sum_{s \in S} v(s)$. For example, if

$$T = [1, 5, 2, 4, 1] \text{ and } P = \{([2], 3), ([1], 1), ([5, 2], 8), ([1, 2], 12), ([1, 5], 2)\},$$

then $S_1 = ([1, 5], [2], [1])$, $S_2 = ([1], [5, 2], [1])$, and $S_3 = ([1], [2], [1])$ are all valid gameplays, with total values 6, 10, and 5 respectively. Note playable word $[1, 2]$ cannot exist in any gameplay of T, since $[1, 2]$ is not a contiguous subarray of T. Given T and P, describe an $O(n + m)$-time algorithm to return a gameplay of maximum total value.

Problem 4. [19 points] **Limited-Unlimited**

Given two sets of integers A and B, a **limited-unlimited sequence** of A and B is any sequence S of integers such that each integer $s \in S$ appears in either A or B, and if s appears in A then s appears at most once in S. Given a target sum m and two disjoint sets A and B, each containing exactly n **distinct positive** integers, describe an $O(nm)$-time algorithm to determine whether m is the sum of any limited-unlimited sequence S of A and B, i.e., $m = \sum_{s \in S} s$.

Problem 5. [19 points] **Office Hour Optimization**

Class 0.660 (Algorithms for Introductions) is holding online office hours to help students on three problems — a, b, and c — in three corresponding breakout rooms. The TAs want to develop a 'Sort Bot' to effectively assign each student to a single room at the start of office hours. Assume there are $3n$ students, where each student i has known nonnegative integer **benefit** a_i, b_i, and c_i for being assigned to the room for problem a, b, and c, respectively.

Describe an $O(n^3)$-time algorithm to determine whether it is possible to assign the students **equally** to the three breakout rooms (i.e., n students to each room) while providing **strictly positive** help to every student, and if possible, return the maximum total benefit to students of any such assignment. Note that the assignment must not assign a student to a room for which they would get zero benefit.

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.