Solution: 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 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) 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: Palindrome Party	1	17
3: Bridge Building	1	19
4: The Beavengers	1	19
Total		60

Name: ______________________________

School Email: ______________________________

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

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

Solution: OK!

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

Solution: OK!

Please solve problems (2), (3), and (4) 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 1. [3 points] **Pseudo-Circling**

Indicate whether the given running times of each of problems (2), (3), and (4) are polynomial or pseudo-polynomial by circling the appropriate word below. **One can answer this question without actually solving problems (2), (3), and (4).** (1 point each)

(a) Problem 2: Palindrome Party Polynomial Pseudo-polynomial

Solution: Polynomial

(b) Problem 3: Bridge Building Polynomial Pseudo-polynomial

Solution: Polynomial

(c) Problem 4: The Beavengers Polynomial Pseudo-polynomial

Solution: Pseudo-polynomial

Problem 2. [17 points] **Palindrome Party**

A string is a **palindrome** if its letters in reverse order form the same string. For example, the string $A =$ `'banana'` is not a palindrome, but ten of the **contiguous substrings** of A are palindromes: substring `'b'` occurs once, `'a'` occurs three times, `'n'` occurs twice, `'ana'` occurs twice, and `'nan'` and `'anana'` each occur once, with $1+3+2+2+1+1 = 10$. Given a string of m letters $A = (a_1, \dots, a_m)$, describe an $O(m^2)$-time dynamic programming algorithm to count how many contiguous substrings of A are palindromes.

Solution:

1. Subproblems
   - $x(i, j)$: 1 if substring from letter a_i to letter a_j is a palindrome, 0 otherwise
   - There are $O(m^2)$ possible substrings, so if we compute all $x(i, j)$, we can just count the ones that are 1 in $O(m^2)$ time
2. Relate
   - If $x(i, j)$ is a palindrome for $j - i > 1$ if and only if $a_i = a_j$ and $x(i + 1, j - 1) = 1$
   - $x(i,j) = \begin{cases} 0 & \text{if } a_i \neq a_j \\ x(i+1, j-1) & \text{otherwise} \end{cases}$ for $i, j \in \{1, \dots, m\}$ and $j - i > 1$
   - Subproblems $x(i, j)$ only depend on strictly smaller $j - i$, so acyclic
3. Base
   - $x(i, i) = 1$, one letter is a palindrome
   - $x(i, i+1) = \begin{cases} 1 & \text{if } a_i = a_{i+1} \\ 0 & \text{otherwise} \end{cases}$, two adjacent letters that are the same are a palindrome
4. Solution
   - Solve subproblems $x(i, j)$ in order of increasing $j - i$
   - Return $\sum_{i=1}^{m} \sum_{j=i}^{m} x(i, j)$, which can be computed naively in $O(m^2)$ time
5. Time
   - # Subproblems is $O(m^2)$
   - Work per subproblem is $O(1)$
   - Total running time is $O(m^2)$ (including $O(m^2)$ to compute solution)

Common Mistakes:

- Did not provide any algorithm for determining whether a substring is a palindrome
- (or claiming a linear algorithm is constant time)
- Failed to account for both even and odd palindromes
- Over-counting number of palindromes

Problem 3. [19 points] **Bridge Building**

The McFields and Hatcoys are rival families who live on opposite sides of a river. The McFields own n houses $M = (m_1, \ldots, m_n)$ along one side of the river, and the Hatcoys own n houses $H = (h_1, \ldots, h_n)$ along the other, with house h_i directly opposite house m_i. Each house a is associated with a positive or negative integer **compatibility** $c(a)$.

The local sheriff wants to build bridges between pairs of houses across the river to increase goodwill. Each house may be connected to at most one bridge, and bridges may not cross[1]. If two houses m_i and h_j are connected by a bridge, **goodwill** will increase by $c(m_i) \times c(h_j)$, the product of their compatibilities. Given the compatibility of each house, describe an $O(n^2)$-time dynamic programming algorithm to determine a placement of bridges that will maximize total goodwill.

Solution:

1. Subproblems
   - $x(i, j)$: maximum goodwill achievable by placing bridges between houses in $\{m_1, \ldots, m_i\}$ and houses in $\{h_1, \ldots, h_j\}$
2. Relate
   - Either there is a bridge between m_i and h_j or not. Guess!
   - If there is a bridge, gain $c(m_i) \times c(h_j)$
   - If there is not a bridge, then at least one of m_i and h_j is not connected to a bridge
   - $x(i, j) = \max\{c(m_i) \times c(h_j) + x(i-1, j-1), x(i-1, j), x(i, j-1)\}$
   - Subproblems $x(i, j)$ only depend on strictly smaller $i + j$, so acyclic
3. Base
   - No increase in goodwill if no bridges can be built
   - $x(i, 0) = 0$ for all $i \in \{0, \ldots, n\}$ and $x(0, j) = 0$ for all $j \in \{1, \ldots, n\}$
4. Solution
   - Solve subproblems $x(i, j)$ in order of increasing $i + j$
   - Solution is $x(n, n)$, the maximum goodwill by placing bridges between all houses
   - Store parent pointers to reconstruct placement of bridges
5. Time
   - # Subproblems is $O(n^2)$
   - Work per subproblem is $O(1)$
   - Total running time is $O(n^2)$

Common Mistakes:

- Forgetting to consider the choice where a bridge is not built
- Using indices into H and M that are not independent of each other (or using too many)
- Not providing sufficient base cases or solution from parent pointers

[1]If a bridge is built connecting houses m_i and h_j, houses m_a and h_b may not be connected by a bridge if Booleans $(i < a)$ and $(j < b)$ are not equal, as the bridges would cross.

Problem 4. [19 points] **The Beavengers**

Varmel Studios is working on a new superhero movie called "The Beavengers". There will be s superheroes starring in the movie. Each superhero $i \in \{1, \ldots, s\}$ has a known positive integer **combat strength** p_i. Let $m = \sum_{i=1}^{s} p_i$ be the total combat strength of all of the superheroes. In this film, the superheroes will turn against each other, so the movie producers would like to split them into two teams of equal size and strength if possible. A team's **size** is the number of superheroes on the team, and its **strength** is the sum total of their combat strengths.

Given the combat strengths of all s superheroes, describe an $O(s^2 m)$-time dynamic programming algorithm to determine whether they can be divided into two teams of **equal size and strength**, where every superhero appears in exactly one of the teams. (Assume that s and m are even.)

Solution:

1. Subproblems
   - $x(i, j, k)$: 1 if some subset of j superheroes from $\{0, \ldots, i\}$ can form a team of strength k, and 0 otherwise, for $i \in \{0, \ldots s\}$, $j \in \{0, \ldots, i\}$, and $k = \{0, \ldots, m\}$
2. Relate
   - Superhero i is used on team or not. Guess!
   - If i is used, team needs one fewer player with strength less by p_i
   - If i is not used, team needs same size and strength
   - $x(i, j, k) = \max \left\{ \begin{array}{ll} x(i-1, j-1, k-p_i), & \text{if } k \geq p_i \\ x(i-1, j, k) & \end{array} \right\}$
   - Subproblems $x(i, j, k)$ only depend on strictly smaller i, so acyclic
3. Base
   - $x(i, 0, 0) = 1$, the vacuous case of zero superheroes needed
   - $x(0, j, k) = 0$, can't add to teams or construct a strength difference with no superheroes
   - $x(i, 0, k) = 0$, can't have a team with positive strength but no members
4. Solution
   - Solve subproblems $x(i, j, k)$ in order of increasing i
   - Solution is whether $x(s, s/2, m/2)$ equals 1
5. Time
   - # Subproblems is $O(s^2 m)$
   - Work per subproblem is $O(1)$
   - Total running time is $O(s^2 m)$

Common Mistakes:

- Subproblems do not keep track of partial strength or size
- Insufficient explanation/description of subproblems
- Inconsistent use of prefixes vs. suffixes
- Applying Partition and checking whether partition has same number of heroes

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.