Name _____

CSCI 332, Fall 2025
Exam 3

Note that this exam has three sections. They are:

1. 2D Dynamic Programming (50 points)

2. Complexity Classes (P, NP, and NP-Completeness) (50 points)

You may use a double sided 3x5 handwritten notecard of notes during the test but no other resources. If you need more space than what is given, develop your solution on scratch paper before copying your final answer to the exam paper.

Good luck!

Section 1 (2D Dynamic Programming)

1. (25 points) In this problem, you will write a memoized algorithm to compute $fizz(1, n)$ based on the following (meaningless) recurrence.

$$
fizz(i, k) = \begin{cases}
0 & \text{if } i = n+1 \text{ or } i = n+2 \\
\infty & \text{if } k = 0 \\
\min\left\{\begin{array}{l} fizz(i+1, k), \\ fizz(i+2, k-1) \end{array}\right\} + A[i, k] & \text{if } A[i, k] \text{ is even} \\
\max\left\{\begin{array}{l} fizz(i+1, k), \\ fizz(i+2, k-1) \end{array}\right\} - A[i, k] & \text{if } A[i, k] \text{ is odd}
\end{cases}
$$

(a) (8 points) An appopriate memoization data structure for this recurrence is a 2D array. Let's call it F. What are the exact dimensions of F? Assume that i values are the rows and k values are the rows.

Rows run $i = $ _____ to $i = $ _____ .

Columns run $k = $ _____ to $k = $ _____ .

(b) (4 points) In what order would you fill in the entries of your memoization structure?

Fill rows in (circle one) ascending order/descending order

Fill columns in (circle one) ascending order/descending order

(c) (13 points) Write pseudocode for the iterative, dynamic programming algorithm to compute $fizz(1, n)$.

```
MemoFizz(A):
    # initialize your memoization array F



    # do we have any base cases?




    # how should we fill in the rest of the array?







    # what should we return to get the final answer?
```

2. (25 points)

Legolas and Gimli are paddling down the River Anduin toward Minas Tirith. Refreshed by their visit to Lothlorien, they would like to fight as many orcs as possible, and they have received advance information about n orc camps along their way, each with $Orcs[i]$ orcs. However, every time they stop to fight orcs, they add $Delay[i]$ days to their journey, and they only were gifted k pieces of Lembas bread from Galadriel and Celeborn. Each day they spend, they each must eat one piece of Lembas bread each, and if they run out, they won't make it to Minas Tirith.

If you were to write a dynamic programming algorithm to help Legolas and Gimli choose which orc camps to attack in order to maximize the number of orcs they fight, you would first need to define the subproblems you will solve.

Assume that you receive as input an array $Orcs[1..n]$ and an array $Delay[1..n]$, as well as an integer k representing the number of pieces of Lembas bread they have.

(a) (8 points) Write the *English definition* of the subproblems you will solve in order to write an algorithm for this problem.

(b) Write the *recursive definition* of the subproblems you will solve in order to write an algorithm for this problem. Be sure to include all necessary base cases!

Section 2 (Complexity Classes)

3. (16 points) For each of the following statements, indicate whether it is True or False. **Assume P $\neq$ NP.**

   (a) Every problem in NP is also in P. T or F

   (b) Every problem in P is also in NP. T or F

   (c) The set NP is the set of problems that cannot be solved in polynomial time. T or F

   (d) If a problem A is NP-Complete, then A is in NP. T or F

   (e) If a problem A is NP-Complete, then A is in P. T or F

   (f) If a problem A is in NP, then there exists a polynomial-time verification algorithm for A. T or F

   (g) If there is a polynomial-time reduction from problem A to NP-hard problem B, then problem A is NP-hard. T or F

   (h) If there is a polynomial-time reduction from NP-hard problem A to problem B, then problem B is NP-hard. T or F

4. (10 points) The *Knapsack* problem is defined as follows: Given a set of n items, each with a weight w_i and a value v_i, determine the maximum value of items that can be placed in a knapsack of capacity W.

   (a) Write a *decision* version of the Knapsack problem.

   (b) Prove that your decision version of the Knapsack problem is in NP.

5. (24 points) In this problem, you will show that the *k-Clique* problem is NP-hard by reducing from the *Independent Set* problem.

(a) The *k-Clique* problem is defined as follows: Given an undirected graph $G = (V, E)$ and an integer k, determine whether there exists a subset of vertices $V' \subseteq V$ such that $|V'| = k$ and every pair of vertices in V' is connected by an edge in E. Draw two graphs with 8 vertices each: one that contains a 4-Clique and one that does not contain a 4-Clique.

(b) You notice that any graph G has an edge-complement G' with the same vertices, but with exactly the opposite set of edges——uv is an edge in G' if and only if uv is not an edge in G. A set of vertices is independent in G if and only if the same vertices define a clique in G'. Thus, a size k independent set in G has the same vertices (and thus the same size) as a clique in G'. Use this observation to write a polynomial-time reduction from the Independent Set problem to the k-Clique problem.