This is a diagnostic exam intended to help you evaluate your readiness for the real exam.

• On the real test, remember to look for your name 😊

• If you believe a question is ambiguous, write at least two reasonable interpretations and indicate clearly which one you will be using. Then answer your question with that assumption. Unless your interpretation makes the problem much more trivial than intended, we will grade your response as if one of us had made that clarification.

• We will only grade responses marked in the space provided for each question.

• The following problems are \( \mathcal{NP} \)-complete:
  
  - SAT Given a boolean formula \( \phi \), is it satisfiable?
    
    * Also 3-Sat, where the formula is a conjunction of disjunctions of size three.
  
  - INDEPENDENT SET: Given a graph \( G \) and an integer \( k \), determine if there is some subset of the vertices \( V' \), \( |V'| \geq k \) such that no two vertices in \( V' \) share an edge.
  
  - VERTEX COVER: Given a graph \( G \) and integer \( k \), determine if there is some subset of the vertices \( V' \), \( |V'| \leq k \) such that each edge is incident to some vertex in \( V' \).
  
  - DIRECTED HAMILTONIAN PATH : does this graph \( G \) contain a simple path that includes every vertex?
    
    * Also undirected Hamiltonian Path
    
    * Also Hamiltonian Cycle (directed and undirected)
  
  - 3-COLOR: Given an undirected graph \( G \), determine whether there is a mapping of vertices in \( G \) to 3 distinct colors such that no edge is monochromatic.
  
  - SUBSET SUM: Given a set \( S \) of numbers, as well as a number \( T \), determine if there is a subset of \( S \) that sums to exactly \( T \).

• It has been great having you this quarter. 😊
1. (3 points) The problem of finding the closest pair of points is as follows. We are given an input of a set of \( n \geq 2 \) points in two dimensions. We want to find which two of these are the closest together (via Euclidean distance). There is an algorithm (many, actually) to solve this with running time \( \mathcal{O}(n^2) \).

   (a) Express that optimization problem as a decision problem.

   (b) Is the decision version of the problem in \( \mathcal{P} \)? Why or why not?

   (c) Is the optimization version of the problem in \( \mathcal{NP} \)? Why or why not?

2. (2 points) The Knapsack problem is as follows. We are given a set of \( n \) items, each with a positive weight and a positive value. We also have some maximum weight \( W \), for some input positive integer \( W \). Our goal is to select a subset of the items such that the total weight is at most \( W \) and we achieve the maximum possible value.

   There exists an algorithm to solve this in time \( \mathcal{O}(nW) \). Is this a polynomial time algorithm, as per the definition from lecture and the reading? Regardless of your answer, give a brief (1-2 sentence) explanation for why.

   If your answer is “no,” give also a running time that would be polynomial for this problem. Such an answer must be a function of both \( n \) and \( W \) (for example, you cannot just answer “\( \mathcal{O}(1) \)” or “\( \mathcal{O}(n) \)”).
Write your answer to question 1(a) between here and the dotted line.

Write your answer to question 1(b) between here and the dotted line.

Write your answer to question 1(c) between here and the dotted line.

Write your answer to question 2 between here and the end of the page.
3. (3 points) The **Four-Sum** problem asks: given a vector of \( n \) distinct integers, each of which may individually be positive, negative, or zero, and a target integer \( T \), is there a subset of at most four elements from \( n \) that add up to \( T \)? Note that this is a special case of **Subset Sum**. There is an algorithm to solve this that takes \( O(n^4) \) time.

Does the existence of that algorithm prove \( P = NP \)? Why or why not? You **DO NOT** need to, nor should you, provide the algorithm described.

4. (2 points) The **Rectangle Packing** problem is as follows. The input is a set of 2-D rectangles in the form of their respective widths \( w_i \) and heights \( h_i \). The input also has a large rectangle with width \( W \) and height \( H \). The **Rectangle Packing** question asks: is it possible to place all of the 2-D rectangles without rotating them so that each is within the perimeter of the larger rectangle but no two overlap? Prove that this problem is \( NP \)-complete. **Hint:** When proving this is in \( NP \), the certificate should include the upper-left corner for each rectangle. When writing your verifier, you may assume you have a function that takes two rectangles as parameters, along with their upper-left coordinates, and determines if they overlap.
Write your answer to question 3 between here and the dotted line.

Write your answer to question 4 between here and the end of the page.