1. For any integer \( k \), the problem \( k\text{Sat} \) is defined as follows:

   • **INPUT**: A boolean formula \( \Phi \) in conjunctive normal form, with exactly \( k \) distinct literals in each clause.
   • **OUTPUT**: \( \text{True} \) if \( \Phi \) has a satisfying assignment, and \( \text{False} \) otherwise.

   (a) Describe a polynomial-time reduction from \( 2\text{Sat} \) to \( 3\text{Sat} \), and prove that your reduction is correct.
   
   (b) Describe and analyze a polynomial-time algorithm for \( 2\text{Sat} \). *[Hint: This problem is strongly connected to topics covered earlier in the semester.]*
   
   (c) Why don’t these results imply a polynomial-time algorithm for \( 3\text{Sat} \)?

2. This problem asks you to describe polynomial-time reductions between two closely related problems:

   • **SubsetSum**: Given a set \( S \) of positive integers and a target integer \( T \), is there a subset of \( S \) whose sum is \( T \)?
   • **Partition**: Given a set \( S \) of positive integers, is there a way to partition \( S \) into two subsets \( S_1 \) and \( S_2 \) that have the same sum?

   (a) Describe a polynomial-time reduction from **SubsetSum** to **Partition**.
   
   (b) Describe a polynomial-time reduction from **Partition** to **SubsetSum**.

   Don’t forget to prove that your reductions are correct.

3. **Pebbling** is a solitaire game played on an undirected graph \( G \), where each vertex has zero or more pebbles. A single **pebbling move** removes two pebbles from some vertex \( v \) and adds one pebble to an arbitrary neighbor of \( v \). (Obviously, \( v \) must have at least two pebbles before the move.) The **PebbleClearing** problem asks, given a graph \( G = (V, E) \) and a pebble count \( p(v) \) for each vertex \( v \), whether is there a sequence of pebbling moves that removes all but one pebble. Prove that **PebbleClearing** is \( \text{NP-hard} \).
Solved Problem

4. Consider the following solitaire game. The puzzle consists of an \( n \times m \) grid of squares, where each square may be empty, occupied by a red stone, or occupied by a blue stone. The goal of the puzzle is to remove some of the given stones so that the remaining stones satisfy two conditions:

(1) Every row contains at least one stone.
(2) No column contains stones of both colors.

For some initial configurations of stones, reaching this goal is impossible; see the example below.

Prove that it is NP-hard to determine, given an initial configuration of red and blue stones, whether this puzzle can be solved.

Solution: We show that this puzzle is NP-hard by describing a reduction from 3Sat.

Let \( \Phi \) be a 3CNF boolean formula with \( m \) variables and \( n \) clauses. We transform this formula into a puzzle configuration in polynomial time as follows. The size of the board is \( n \times m \). The stones are placed as follows, for all indices \( i \) and \( j \):

- If the variable \( x_j \) appears in the \( i \)th clause of \( \Phi \), we place a blue stone at \((i,j)\).
- If the negated variable \( \overline{x_j} \) appears in the \( i \)th clause of \( \Phi \), we place a red stone at \((i,j)\).
- Otherwise, we leave cell \((i,j)\) blank.

We claim that this puzzle has a solution if and only if \( \Phi \) is satisfiable. This claim immediately implies that solving the puzzle is NP-hard. We prove our claim as follows:

\[ \implies \] First, suppose \( \Phi \) is satisfiable; consider an arbitrary satisfying assignment. For each index \( j \), remove stones from column \( j \) according to the value assigned to \( x_j \):

- If \( x_j = \text{TRUE} \), remove all red stones from column \( j \).
- If \( x_j = \text{FALSE} \), remove all blue stones from column \( j \).

In other words, remove precisely the stones that correspond to \( \text{FALSE} \) literals. Because every variable appears in at least one clause, each column now contains stones of only one color (if any). On the other hand, each clause of \( \Phi \) must contain at least one \( \text{TRUE} \) literal, and thus each row still contains at least one stone. We conclude that the puzzle is satisfiable.
On the other hand, suppose the puzzle is solvable; consider an arbitrary solution. For each index $j$, assign a value to $x_j$ depending on the colors of stones left in column $j$:

- If column $j$ contains blue stones, set $x_j = \text{TRUE}$.
- If column $j$ contains red stones, set $x_j = \text{FALSE}$.
- If column $j$ is empty, set $x_j$ arbitrarily.

In other words, assign values to the variables so that the literals corresponding to the remaining stones are all \text{TRUE}. Each row still has at least one stone, so each clause of $\Phi$ contains at least one \text{TRUE} literal, so this assignment makes $\Phi = \text{TRUE}$. We conclude that $\Phi$ is satisfiable.

This reduction clearly requires only polynomial time.

\[\square\]

**Rubric (for all polynomial-time reductions):** 10 points =

+ 3 points for the reduction itself
  
  - For an NP-hardness proof, the reduction must be from a known NP-hard problem. You can use any of the NP-hard problems listed in the lecture notes (except the one you are trying to prove NP-hard, of course).

+ 3 points for the "if" proof of correctness

+ 3 points for the "only if" proof of correctness

+ 1 point for writing "polynomial time"

• An incorrect polynomial-time reduction that still satisfies half of the correctness proof is worth at most 4/10.

• A reduction in the wrong direction is worth 0/10.