Question 1: Proving NP Completeness
An undirected graph \( G \) is \( k \)-colorable if there is a way to color its vertices with \( k \) colors, such that two adjacent nodes get different colors.
Formally: \( G = (V, E) \) is \( k \)-colorable if there exists a map \( c : G \to \{1, \ldots, k\} \) such that for all edges \( (u, v) \in E \) it holds that \( c(u) \neq c(v) \).
Let \( k\text{-COL} = \{ G : G \text{ is } k\text{-colorable} \} \) be the language of \( k \)-colorable graphs.

- (a) Prove that 3-COL is NP complete.
- (b) Prove that 2-COL is in P.

Question 2: Collapses of the polynomial hierarchy
Prove that if for some \( i \geq 1 \) it holds that \( \Sigma_i = \Pi_i \), then \( PH = \Sigma_i = \Pi_i \), that is the polynomial hierarchy collapses to the \( i \)-th level.

Question 3: Co-NP Completeness
Recall that:
1. A language \( L \) is in coNP if its complement \( L^c \) is in NP.
2. A language \( L \) is coNP-hard if for any language \( L' \) in coNP, there is a poly-time reduction from \( L' \) to \( L \).
3. A language \( L \) is coNP-complete if it is both in coNP and coNP-hard.

Prove that \( L \) is coNP-complete iff its complement \( L^c \) is NP-complete.

Question 4: Designing algorithms in logspace
Consider the language \( SUMEQUAL = \{ x\#y\#z : x, y, z \in \{0, 1\}^*, x + y = z \} \). Here, we consider \( x, y, z \) as integers represented in binary, and \# is a special character that separates them. Prove that \( SUMEQUAL \) is computable in logarithmic space (that is, \( SUMEQUAL \in L \)).
**Question 5:**
Recall the NL-complete language CONN:
CONN=\{(G,s,t): G is a directed graph, s,t are nodes in G, there is a path in G from s to t\}.

Assume G has n nodes. There are two families of algorithms to solve CONN:
1. BFS/DFS based algorithms. These use \(O(n)\) space and polynomial time (concretely \(O(|E|) = O(n^2)\) time).
2. Savitch’s algorithm which uses \(O(\log^2 n)\) space.

(a) How much time does Savitch’s algorithm need? Does it to run in polynomial time? Why or why not? Hint: express the asymptotic time complexity of Savitch.
(b) If you restrict your algorithm to run in poly-time, what is the minimal amount of space you can achieve? Can you beat the linear space used by BFS/DFS?

There is no “textbook solution” for this question. Instead, I want to see your best effort and creative ideas.