**Problem 1:**

1. True
2. False
3. False
4. False
5. True
6. False
7. False
8. True

**Problem 2:**

1. In order to prove that the 4-SAT problem is NP-Hard, deduce a reduction from a known NP-Hard problem to this problem. Deduce a reduction from which the 3-SAT problem can be reduced to the 4-SAT problem. For each clause of the 3-SAT formula f, for example, a literal a and its corresponding complement a’ should be added to the formula. Let there be a clause c, such that c = u V v’ V w . To convert it in 4-SAT, we convert c to c’, such that,    
   c’ = (u V v’ V w V a) AND (u V v’ V w V a’).   
   After simulating this conversion, two properties hold :

* If 3-SAT has a satisfiable assignment, which means, every clause evaluates to true for a specific set of literal values, then 4-SAT will also hold, because each clause-set is formed by a combination of a literal and its complement, whose value won’t make any difference.
* If 4-SAT is satisfiable for any (u V v V w V a) and (u V v V w V a’), then 3-SAT is also satisfiable because a and a’ are complement, which indicates that the formula is satisfiable due to some other literal except a too.

Therefore, following the above propositions, the 4-SAT problem is NP-Complete.

1. The algorithm is as follows:

|  |
| --- |
| Init a an undirect graph G with (V,E), and initial all vertices are all false.  For item\_i in vertices in CNF   * If item\_i is true then change vertical\_i in G to true   If All vertices in G are all true then return true  Else return false. |

**Problem 3:**

**Algorithm Describe:**

 Incrementally add to $G$ in such a way that at each step we learn the coloring of one vertex. Here we add a triangle of vertices (so they must all be different colors) and then, for each vertex, try connecting it to every combination of 2 of those new vertices (so its color will be the same as the new vertex it is not connected to). The algorithm is as follows:

|  |
| --- |
| If G is not 3-colorable then return null  G <- G with new vertices r, g, b and edges (r, g), (g,b), (b,r)  For each vertex v\_i   * G\_r <- G’ with edges (v\_i, g), (v\_i, b) * G\_g <- G’ with edges (v\_i, r), (v\_i, b) * G\_b <- G’ with edges (v\_i, r), (v\_i, g) * G’ <- whichever of G\_r, G\_g, G\_b is 3-colorable   For each vertex v, the color of v is the one of r, g, b it is not connected to in the final G’. |