CS 21 Problem Set #5
Ishaan Mantripragada

Problem 1

a.
We will show that the language 2-COLORABLE is in P by reducing it to 2-SAT which we know is in P. We can build a graph G as follows: Every vertex in G can have 2 possible colors. For every pair of vertices (u, v) that is connected by an edge, if vertex u is color c_1, then v must be color c_2. This can be rewritten using boolean clauses as follows: For every pair of vertices (u, v) that is connected by an edge, we can create the clauses $(u \vee v) \wedge (\neg u \vee \neg v)$. This reduction is a polynomial time operation. Now we must show that both yes maps to yes and no maps to no for this reduction.

Lets suppose that we have a satisfying assignment to the boolean clauses. If this is the case, it follows that no adjacent vertex is assigned to the same color. If we let 'True' map to one color and 'False' map to the other color, we build a graph G such that every vertex of one color is not connected to any other vertex of the same color. Additionally, there are only two colors in the graph. Therefore, G has a valid 2-coloring.

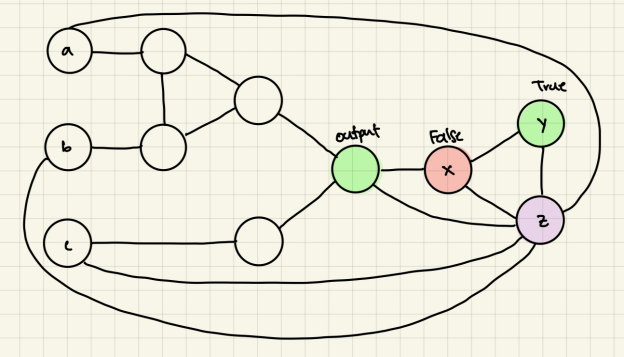
Now, lets suppose that G is 2-colorable. Then, we can assign every vertex of one color to 'True' and every vertex of the other color to 'False'. Since G is 2-colorable, every pair of vertices connected by an edge will have opposite values. Thus, the clauses $(u \vee v) \wedge (\neg u \vee \neg v)$ hold.

Since we have shown a valid reduction from 2-COLORABLE to 2-SAT, it follows that 2-COLORABLE is in P.

b.

To show that 3-COLORABLE NP-complete, we must show that it is in NP and is NP-hard. We know 3-COLORABLE is in NP because we can simply go through all nodes and edges in the graph to confirm that no edge connects two nodes of the same color. This occurs in polynomial time.

Now, to show that 3-COLORABLE is NP-hard, we can reduce it from 3-SAT. Let us consider some instance ϕ of 3-SAT. We can create the following graph.



The base of the graph is the triangle of three colors on the far right (x, y, z). These three nodes are the 'starting' nodes and force other nodes to be certain colors for the graph to be 3-colorable. The three nodes on the far left (a, b, c) represent the literals for each clause ($a \vee b \vee c$). Now for the graph to be a valid reduction from 3-SAT, we can assign one of the colors to 'True' (Green) and one of the remaining colors to 'False' (Red). Since we only want the output vertex to evaluate to 'True', we only connect it to the nodes in the base that are not 'True'. Additionally, we also only want a, b, and c to be either 'True' or 'False', so we connect them to the node in the base that was not assigned to either 'True' or 'False'. From this construction, we can see that a properly colored graph results in one of a, b, or c being the same color as the output node in the center (at least one of the variables must be true). Now we must show that both yes maps to yes and no maps to no for this reduction.

Lets suppose that we have a satisfying assignment ϕ. If our base nodes are colored as shown in the figure, then at least one of a, b, or c will be colored true and the output node will be colored true. We can add as many clauses as necessary to represent 3-SAT. Therefore, we shown a valid 3-coloring.

Now, lets suppose that G is 3-colorable. If G is 3-colorable, it follows that our base nodes are all different colors. We can assign x to False, y to True, and z to some arbitrary value. If we continue with the 3-coloring for the rest of the graph, at least one of a, b, or c will be colored 'True' and the output node will also be colored 'True'. This coloring can be done for every possible clause. Thus, we have shown a satisfying solution.

Since we have shown a valid reduction from 3-COLORABLE to 3-SAT, it follows that 3-COLORABLE is NP-complete.

Problem 2 (grade completely)
To show that (3,3)-SAT NP-complete, we must show that it is in NP and is NP-hard. We know (3,3)-SAT is in NP because we can simply go through all nodes and edges in the graph to confirm the restrictions. This occurs in polynomial time.

Now, to show that (3,3)-SAT is NP-hard, we can reduce it from 3-SAT. Let us consider some instance ϕ of 3-SAT with n variables. We can construct an instance ϕ' of (3,3)-SAT that has at most 3 variables. For every variable x that occurs more than 3 times, we can replace it with $(x_1, x_2, x_3, ...x_{j-1}, x_j)$ where j is the number of occurrences in ϕ. We can then add the clauses $(\neg x_1 \vee x_2) \wedge (\neg x_2 \vee x_3) \wedge ... \wedge (\neg x_{j-1} \vee x_j) \wedge (\neg x_j \vee x_1)$. In this replacement, we can see that every variable occurs only twice.

For every satisfying argument, $x_1, x_2, ..., x_j$ must have the same value. Any solution to ϕ can be made into a solution to ϕ' by setting every $(x_1, x_2, ..., x_j)$ equal to the original x that it replaced which occurred more than 3 times. Also, any solution to ϕ' can be made into a solution to ϕ by setting x to x_1 for every x occurring more than three times.

Since we have shown a valid reduction from (3,3)-SAT to 3-SAT, it follows that (3,3)-SAT is NP-complete.

Problem 3

Let us consider the 10 clauses provided in the hint:

$(x \vee x), (y \vee y), (z \vee z), (w \vee w), (\neg x \vee \neg y), (\neg y \vee \neg z), (\neg z \vee \neg x), (x \vee \neg w), (y \vee \neg w), (z \vee \neg w)$

Exactly one of x, y, or z is true
If x is true, y is false, z is false, and w is true, then 6 of the clauses are satisfied. If x is true, y is false, z is false, and w is false, then 7 of the clauses are satisfied. (The same applies for if y or z was the only true variable).

Exactly two of x, y, or z is true
If x is true, y is true, z is false, and w is true, then 7 of the clauses are satisfied. If x is true, y is true, z is false, and w is false, then again 7 of the clauses are satisfied. (The same applies for any combination of x, y, and z).

x, y, and z are true
If x is true, y is true, z is true, and w is true, then 7 of the clauses are satisfied. If x is true, y is true, z is true, and w is false, then again 7 of the clauses are satisfied.

x, y, and z are false
If x is false, y is false, z is false, and w is true, then 4 of the clauses are satisfied. If x is false, y is false, z is false, and w is false, then 6 of the clauses are satisfied.

Thus, any combination of assignments for x, y, z, and w can only result in a maximum of 7 of the 10 clauses to be satisfied. We can use this information for the rest of the proof.

To show that MAX2SAT NP-complete, we must show that it is in NP and is NP-hard. For MAX2SAT, given a 2-CNF formula and an assignment of variables, we can verify whether at least k clauses are satisfied by simply substituting the values of the variables into the formula and checking each clause, which can be done in polynomial time. Thus, MAX2SAT is in NP. To show that MAX2SAT is NP-hard, we can reduce it from 3-SAT. Let us consider some instance ϕ of 3-SAT. We can create an instance ϕ' of 2-CNF such that at least k clauses of ϕ' can be satisfied if and only if at least k clauses of ϕ can be satisfied. In ϕ', each clause is one of the 10 clauses above and $k = 7n$ (where n is the number of clauses) since we have shown that only a maximum of 7 clauses can be satisfied. This reduction can be done in polynomial time.

Lets suppose that we have a satisfying assignment ϕ. Since we have shown that at most 7 out of the 10 clauses can be satisfied with an assignment, the solution will satisfy at least $7n$ of the clauses in ϕ'. Now, lets suppose that we have a solution that satisfies at least $7n$ clauses in ϕ'. Utilizing the fact that we can only satisfy at most 7 of the 10 clauses with an assignment, it follows that in every group of 10 clauses, we must have 7 that are satisfied.

Since we have shown a valid reduction from MAX2SAT to 3-SAT, it follows that MAX2SAT is NP-complete.