Question 2

(a) For this algorithm, we can bipartite color the tree so that every node has a different color than its children. We can then assign each color to either next week (1) or the week after that (2) so that every supervisor has a different week off than their direct reports. This algorithm accurately makes it so that no one has this week off, so we have minimized the number of 0's, and bipartite coloring of the tree ensures that the necessary relationships hold.

(b) Let T(n) represent the type of vacation for the person at node n (0 being this week off, 1 being next week off, 2 being the week after next off).

$$T(n) = \begin{cases} X, & \text{if pre-labelled as X for 0, 1, or 2.} \\ -\infty, & \text{if P-BASED(n) } \cap \text{ C-BASED(n) DNE} \\ MAX(P-BASED(n) \cap \text{ C-BASED(n)}), & \text{otherwise} \end{cases} \tag{1}$$

$$C{-}BASED(n) = \begin{cases} \{0,1,2\} - \{T(CHILDREN(n))\}, & \text{for children of n with a single possible value.} \\ \{0,1,2\}, & \text{if no children of n have a single possible value} \end{cases} \tag{2}$$

$$P{-}BASED(n) = \begin{cases} \{0,1,2\} - \{T(PARENT(n)))\}, & \text{if the parent of n has a single possible value.} \\ \{0,1,2\}, & \text{otherwise} \end{cases} \tag{3}$$

CHILDREN(n) and PARENT(n) refer to the parent and child nodes of n respectively. C-BASED(n) will return the possible values that can occur for a node based on the values of its children. Similarly, P-BASED(n) will return the possible values that can occur based on the values of its parent. T(n) calculates the final value at node n by taking the maximum possible value found in both C-BASED and P-BASED so that 0 has the lowest priority. To get the total number, we have to iterate through the entire tree once after calculating all T(n) and count the number of 0s. We can initiate this recurrence by calling T(root) and then calling the final post order traversal while keeping track of the count.

(c) No it is not because we will not be able to ensure that a node complies with both its parents and its children with a post order only. We need both a post-order and a pre-order traversal to do this. This means we can use a post-order first to ensure every node complies with its parents, then a pre-order to ensure every node complies with its children, and finish by using another post-order traversal to count the total number of 0s in the tree.

(d) Since we will need to travel through every node for both the pre-order and post-order traversals, this will run in O(n) time.

Question 3

(a) Let a_5 represent the total number of photos labelled by Company A (all cost 5 cents each).

Let b_3 represent the total number of photos labelled by Company B that cost 3 cents.

Let b_4 represent the total number of photos labelled by Company B that cost 4 cents.

Let b_7 represent the total number of photos labelled by Company B that cost 5 cents.

(b) $-a_5 <= 0$ and $a_5 <= 400$

$-b_3 <= 0$ and $b_3 <= 200$

$-b_4 <= 0$ and $b_4 <= 700$

$-b_7 <= 0$

$a_5 + b_3 + b_4 + b_7 <= 1000$ and $-(a_5 + b_3 + b_4 + b_7) <= -1000$

Minimize $(5a_5 + 3b_3 + 4b_4 + 7b_7)$ subject to the above constraints.

(c) We will add a variable a_4 that represents the total number of photos labelled by company A that cost 4 cents.

New constraints for equation one:

$-a_5 <= 0$ and $a_5 <= 100$, remove $a_5 <= 400$.

$a_5 + b_3 + b_4 + b_7 <= 1000$ and $-(a_5 + b_3 + b_4 + b_7) <= -1000$

Equation one: $(5a_5 + 3b_3 + 4b_4 + 7b_7)$

New constraints for equation two: $a_5 <= 100$ and $-a_5 <= -100$ and $-a_4 <= 0$ and $a_4 <= 300$, remove $-a_5 <= 0$ and $a_5 <= 400$.

$a_5 + b_3 + b_4 + b_7 + a_4 <= 1000$ and $-(a_5 + b_3 + b_4 + b_7 + a_4) <= -1000$

Equation two: $(5a_5 + 4a_4 + 3b_3 + 4b_4 + 7b_7 + 4a_4)$

Now we have to both minimize both equation one and two and then take the smallest value between the two as the total minimum.