Question 2

1. **Algorithm**
   In the first round each node takes color "ID+1" and algorithm finishes.
   1 is $O(1)$ (constant).
   Proof
   Each node has different from others color (as IDs were different), therefore its proper coloring. Each node can do this, because initially colors are 1 and needed colors are from 1 to n (as IDs are from 0 to $n-1$)

2. It's possible.
   Algorithm
   Start from the node with ID 0. It changes color to 2. Other nodes as soon as observe change of the color to c of one of their neighbors, change color to $5-c$ and stop.
   Proof
   As it's tree, only one neighbor of the node can change color in one round (otherwise there should be a cycle from two paths from this node to the 0–node), therefore color is changed deterministically. It will take no more than n steps (which is $O(n)$, as in each round at least one node changes its color (until the end of propagation). In the end all nodes have colors 2 and 3 (3–max–coloring) as they are changing color observing change of neighbor $1 \to 2$ or $1 \to 3$ and change themselves to the complementary to 5 color. It is a proper coloring, as color of the node and its previous and next in propagation nodes' colors are complementary to 5.

3. It's not possible in principle to create such an algorithm under these conditions.
   Proof
   As soon as one node changes its color (to 2) it can't transmit any information further after that (as we need to have 2–max–coloring, so it can't influence any other node decision after that, because its neighbors can't change their color which is the only way to transmit information). So neighbors of its neighbors (except it itself) can't get any information of what color they should be, as, for example, in linear tree graph (path) the color of ends should be coordinated somehow, but in this situation information of the color of the end can't go further than its one neighbor.

4. **Algorithm**
   Each node converts its ID to binary system complementing it with zeros from the left until its length is $\lceil \log n \rceil$ (let't call it *bin*). Then for $\lceil \log n \rceil$ rounds each node goes through *bin* and increases its color by the corresponding digits, one digit per round. Every other node can store the sequence of colors of its neighbors. After $\lceil \log n \rceil$ rounds each node calculate round differences of colors of its neighbors and transforms resulted sequence of zeros and ones from binary to decimal system, so gets ID.
   Proof
   Every ID fits to $\lceil \log n \rceil$ binary number, as the maximal ID is $n-1$. Number of rounds is $\lceil \log n \rceil + 1$, one for change corresponding to $\lceil \log n \rceil$ digits and one for the final retrieval of colors of neighbors. It's $O(\log n)$.
   Throughout the algorithm no color is more than $\lceil \log n \rceil + 1$, as it can't increase more than by $\lceil \log n \rceil$ from 1, when increases correspond to digits of binary number of the length $\lceil \log n \rceil$.

5. **Algorithm + proof**
   Peeling
   Every node knows from the beginning number of its neighbors. Using an idea of "peeling" from the chapter 1.3 of the lecture book [1], they start to split to sets in such a way: first nodes with degree no more than 2 goes to the set L_1 and inform others by increasing their color by 1 (other nodes don't increase their color at this step). Then, every other node calculates how many neighbors it has excluding already included to some set L_j ($j \leq i$, at i–th step), and if this

number is at most 2, it joins the set L_{i+1}. The number of these sets is no more than $\log_{1.5} n$, because at every step all not included to any set L nodes form a forest F_i with no more than $n-1$ edges (as we had a tree in the beginning and just deleted some nodes), therefore number of nodes with degree at least 3 in F is no more than $\frac{2|F_i|-1}{3} < \frac{2|F_i|}{3}$. It means that number of not included to any L-set nodes is decreasing exponentially and after no more than $\lceil \log_{1.5} n \rceil$ rounds all nodes are distributed between L–sets and this procedure takes $O(\log n)$ (that's the number of sets L).
As all the nodes could count at which step their neighbors increase their color, every node now knows sets of itself and all its neighbors. So at the step $\lceil \log n \rceil + 1 + \lceil \log_{1.5} n \rceil + 1$ (which is still $O(\log n)$) they start the next phase.

3–coloring
So now every node knows if neighbors are from the same peeling set L or not. Inside these L (where all nodes have degree 0, 1 or 2) we can do 3–coloring in $O(2 \log 2 + \log^* n) = O(\log^* n)$ (as in Theorem 1.24 from the same book (and lectures)), which is even less than $O(\log n)$ rounds. In order to comply with the setting (colors can't decrease) nodes first communicate their IDs to their neighbors in $N = \lceil \log n \rceil + 1 = O(\log n)$ as in previous subtask, then choose one of their neighbor as a parent (other, if exist, would be a child) and do algorithm from Theorem 1.4 [1] changing the way color is communicated — now they need no more than N rounds to communicate their new color using the same method from previous subtask, and as in every color–changing round the color of the node increases by no more than $O(N \log^* n) = O(\log^2 n)$, colors still are in $O(\log^2 n)$. After that all nodes from all L sets have their color from 1 to 3 and can set is the same among all sets e. g. $\log^3 n$, $\log^3 n + 1$ and $\log^3 n + 2$ (to be sure that it's possible to achieve these by increasing, order of $\log n$ can be reduced if matters) corresponding to 1, 2 and 3. Therefore, using modification of the locally–directed algorithm from the Theorem 1.4 we color all sets L in 3 colors.

Gluing
Following again Theorem 1.4 we start from $L_{i_{max}}$ (maybe empty), where $i_{max} = \lceil \log_{1.5} n \rceil + 1$ and consequently merge it with $L_{i_{max}-1}, L_{i_{max}-2}$ and so on preserving effective 3–coloring in merged set, such as all other nodes just increase their color by 1 every L–merging–round, but nodes in merged sets follow the color–schedule in $\{1, 2, 3\}$ and pick the appropriate color knowing colors of neighbors from already merged set and new colors of neighbors from its set who did color–scheduling earlier. Change of color is achieved by adding to in order to achieve $\log^3 n + 3j$, $\log^3 n + 3j + 1$ and $\log^3 n + 3j + 2$ corresponding to 1, 2 and 3, and where $j = i_{max} - i_{current}$, $i_{current}$ corresponds to the index of L set which is merging at this round. And this change also serves as new color–communication to the neighbors.
After getting through all the sets L which number is $O(\log n)$, we will spend $3 * O(\log n) = O(\log n)$ rounds and color will be changed by $3(O(\log n + 1) = O(\log n)$, so at the end we'll have effectively 3–coloring (3 distinct colors used) with colors no more than $O(\log^3 n)$ and using overall no more than $O(\log^2 n)$ rounds. To comply with the task if we take $c = 3$ then our number of rounds is in $O(\log^2 n)$ which is in $O(\log^4 n) = O(\log^{c+1} n)$, what was asked. Therefore, we are done.

References

[1] M. Ghaffari, "Distributed graph algorithms." `https://disco.ethz.ch/courses/fs19/podc/lecturenotes/LOCAL.pdf`.