Question 3

Algorithm
Every node gets to know IDs of neighbors of its neighbors in one round. Also prepares array to check if neighbor (and the node itself) includes to the final graph or not (initially all are "included").
Then, we can color G in $\Delta + 1$ colors using Kuhn-Wattenhofer algorithm (Theorem 1.24, [1]) in $O(\Delta \log \Delta + \log^* n)$.
Including/deleting
We go sequentially through colors and every node checks if it's state is "included" and it participates in some triangles (can do as knows IDs and states of neighbors). If both are true, it changes its state to "not included" and notify its neighbors. If it's state is "not included" and after changing it to "included" it doesn't introduce new triangles, it changes state to "included". After decision node notifies its neighbors of its current state.
Proof
As after each round, number of triangles (if they are) can't increase (including and deleting of node in this algorithm doesn't increase number of triangles, but whenever there is a triangle, at step of the color of one of its vertices it will disappear), the algorithm will result sometime in zero number of triangles, and then after one "all–round" (doing through all the colors) we'll be done (no triangles, no vertex to add).
But we can see, that even one "all–round" is enough to delete all the triangles (as if there is a triangle, at the step of one of its node's color it will be destroyed), so overall we need only two "all–rounds" with at most $2(\Delta + 1)$ real-time rounds: one round to be sure to destroy triangles and one round to include some vertices if it could be done without creating triangles.
And the graph will be as needed: without triangles (as they were deleted during the first round) and we can't add anything (as all possible nodes were added during the second round, it's not possible that some other nodes could be added after that, as adding of the node can't help other nodes who were not able to be added before, because adding of the node only introduces new edges and therefore if including of some other node introduced triangle, it would do that even after including of any other node).

Therefore, overall we will need $O(\Delta \log \Delta + 2\Delta) + O(\log^* n)$ rounds and $f(\Delta) = O(\Delta(2 + \log \Delta)) = O(\Delta \log \Delta)$.

References

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