1: Consensus in 2 processor round-by-round adversary model with constraints on adversary

Analysis:
The first part is trivial, if a processor hears something then output that and the processor doesn't hear anything outputs its own bit. The problem considers solving consensus when the two processors know that at some future round the adversary will delete a message on the link between them. This guarantee breaks the symmetry and thus the processors will now have more information that they started off with once this round occurs. In particular, in the round that one processor doesn't receive a message from the other - it can safely declare itself to be the 'leader' i.e. the final output will either be its bit or a function computed on both processors' bits. Moreover it is interesting to notice that in any round if a processor receives a message containing its own id - it can exit knowing safely that either the other processor received the same bit or that the other processor has agreed to the same bit, either case consensus will be achieved.

Algorithm: In the round immediately after a message is deleted, the 'leader' sends a message indicating that it has reached a consensus by agreeing to its own bit but doesn't output the value of this consensus yet. For the follower (a processor that doesn't believe itself to be the leader yet), there are now two scenarios:

It receives a message This case is trivial as the follower hears the leader tell it the final answer and simply outputs it and continually sends this to its partner.

It doesn't receive a message This is more interesting as the follower now thinks it's the leader and will do the same as the original leader, therefore it will send its own bit with a message indicating the belief that it is the leader itself.

For the leader, there are now four scenarios:

It receives a message containing its own id It simply outputs this as this means that the follower has already agreed to the the leader's bit.

It receives a message simply containing the follower's id (not the same as its own) It does nothing in this round and continues sending the same message as before - a message indicating its agreement to its own bit and goes on to the next round (where this scenarios cannot exist - hence it must output an answer in the next round).

It receives a message from the processor that it thought to be a follower claiming to be a leader But since this processor believes itself to be the leader and then immediately heard this, therefore it didn't receive the other processor's message in the previous round (but thus the other one did) and hence it knows that the true leader is the other processor (as it became a leader earlier) and hence it simply agrees to the message and outputs the bit indicated while continuing to echo back the bit that was agreed to.

It doesn't receive a message This case is also trivial, as now the leader knows that the follower received the bit it agreed to and hence it simply outputs this bit.

Running time:
If the adversary deletes a message in the kth round, this algorithm will run for at most k + 2 rounds.

2: Solving classical wait-free adversary model on stronger model where adversary can delete two messages on a link such that for every P_i and P_j at least one's messages can still reach the other

---

Analysis:
Since the latter is a generalization of the latter, it is not immediately obvious that the latter can possible solve the former as there will be cases where a processor will not hear about any new processors but this doesn't mean that it can output its set (as it could in the trivial solution for the former task on the former model of computation)

Algorithm: Since the constraint on this generalization is that while both links between P_i and P_j can be deleted there must still be a path in the induced graph from either P_i to P_j or vice-a-versa. Moreover, since it is known by all processors that at most n processors will participate in this task, it is known that the maximum length of such a path is n -1. Using this understanding the algorithm emerges:

At the beginning of the task, set your counter to 0
for every round until you output **do**
Send as a message to everyone all the ids you have received as well as your own
if You receive a new id in this round **then** Reset your counter and add this to the list of ids to broadcast
elseIncrement counter by 1
if Counter has reached n - 1 or your set has n processors **then** Output your set and now simply keep broadcasting ids for the rest of the computation

Correctness:
This is algorithm is correct as in every round progress is made: either towards n - 1 rounds of 'silence' i.e. no new ids heard or towards a set that contains all ids. Moreover, there is a stopping condition in both cases, therefore the algorithm will make progress and will output for each processor. Furthermore it is easy to see in this way, that every processor
$P_i either \in S_j or P_j \in S_i or both$ as there must be a path from one to the other at least and and in n - 1 rounds, this path must be completely traversed.

3: Bonus Question: Prove in the round by round adversary model, there is at least one 'king' processor in 2 successive rounds

Analysis:
Using the observation that the problem is significantly easier in the case where the behavior of the adversary is the same in the 2 rounds, the first step was to show that there is a 'king' processor in this case. Notice that the processors that the king processor hears back from are simply the vertices that have a path of at most 2 edges to the vertex representing the king processor in the induced graph Proof by contradiction:
Let v be the vertex with largest in-degree in this graph and let N be the set of vertices that have an edge to v

Assume to the contrary that there is a vertex x that v doesn't 'hear' about it $\implies$ (x, v) doesn't belong to E (the set of edges in the induced graph) but this implies the edge (v, x) must exist.

Moreover, for any w $\in$ N, there doesn't exist (x, w) therefore the edge (w, x) must exist for every vertex w $\in$ N.

But this implies that x has in-degree of size greater than that of v. But this is a contradiction as v is the vertex with greatest in-degree.

Hence the vertex with highest in-degree must necessarily hear about all other vertices, hence there must exist a king processor when the behavior of the adversary is the same in both rounds.