Exercise 1  

1. First, we show that \( r_2(C_{2k+1}) \geq k + 2 \). 
   For this, remember that in each communication step, the edges that messages are transferred over form a matching. As the number of nodes in the cycle is uneven, there must be one inactive node in each such step which doesn’t communicate with any other. W.l.o.g., we name the nodes \( v_1, \ldots, v_{2k+1} \) moving around the cycle in clockwise direction. Assume \( v_1 \) is inactive in step 1. Then, its information has to be transferred to all other nodes, especially the farthest-away ones \( v_{k+1}, v_{k+2} \), starting in step 2. As both these nodes are of distance \( k \) to \( v_1 \), we need at least \( k \) steps to reach one of them, for example \( v_{k+1} \). But now, the other one is not yet having the information: It cannot have started moving to both sides of the cycle at the exact same time (matching!). Thus, we need at least another step to inform \( v_{k+2} \).

2. Now it remains to prove that \( r_2(C_{2k+1}) \leq k + 2 \). 
   Consider the following algorithm:
   For each communication step \( i = \{1, \ldots, k + 2\} \), let \( v_i \) be inactive and let all other nodes communicate along the maximal matching that results from sending over edges \( (v_{i+1}, v_{i+2}), (v_{i+3}, v_{i+4}), \) and so on.
   We prove that after using this algorithm, each vertex in \( C_{2k+1} \) will be having each other vertex’s information. For this, observe that each message travels on in each direction of the cycle in each but the first communication step, except it is waiting right in front of the currently inactive node. This is due to the fact that for the rest of the cycle, two maximal matchings are used in an alternating way. Say a message travels in both directions in some time step if the number of nodes which know of it is enlarged by two.
   Now, assume that the information from a vertex \( v_j \) does not travel on in some step \( i \neq 1 \), or at least not in both directions. The only case in which a message doesn’t travel at all is the one of \( v_1 \) in the first step. In all other cases, it will travel to at least one direction, but possibly not both. This can obviously be the case for messages stuck in the node \( v_j \), \( v_{i-1} \) or \( v_{i+1} \) only. Have a look at the first time after step 1 the message from some node \( v_j \) doesn’t travel in both directions.
   Assume it doesn’t travel clockwise, i.e. it is stuck in \( v_{i-1} \) (because it can’t have reached \( v_{i-1} \) in the previous step, and it would travel on if it was in \( v_{i+1} \)). Then, because in the next step, \( v_{i+1} \) is inactive, it will travel on again. As both the message and the inactive node move forward by one node in each one step, the message will never reach one of the critical positions again and therefore will never have to wait again. Now assume it doesn’t travel counter-clockwise, i.e. it is stuck in \( v_i \) (because it cannot have reached \( v_{i+1} \) in the previous step and it would travel on if it was in \( v_{i-1} \)). In the following step, the message will move on and never have to wait again because node \( v_{i+1} \) will be inactive. So, in all but at most two steps of the algorithm, each message travels into both directions. None of the nodes has distance more than \( k \) from \( v_j \), and therefore after \( k + 2 \) steps, all nodes have the information.
   All in all, the algorithm runs in \( k + 2 \) steps and after that, each of the nodes’ information is known to all others.

Exercise 2

Exercise 2.1  

We first prove that the minimum broadcast time for such a tree results for a broadcast started from the root (or any child of the root).
   Assume the initiating node is resided in one of the root’s subtrees, say, \( T_1 \). Then obviously, to inform all other subtrees, it has to travel through the root (because the path between two nodes is unique in a tree). From there on, the time to inform the whole subtree is still needed, and therefore, the last subtree to be informed by the root will also be the last to finish the broadcast task. Therefore, it is obviously optimal to start at the root.
   We know know that \( \text{minb}(T_k(m)) = b(T_k(m), r) \). It remains to prove that the time needed to perform broadcast from the root is indeed \( mk \) steps.
   For this, have a look at the tree’s levels, the root being on level 0, its children on level 1, and so on. Then, the last node to be informed on level 1 does not have the information until step \( k \), because the root can only inform one child at a time (and it is obviously a good thing to do so, being idle won’t help...). This node now informs the last of its children (on level 2) no earlier than step \( 2k \). On level 3, the last node is informed in step \( 3k \), and so on. As the tree
has \( m \) levels, performing broadcast from the root needs time \( mk \). The algorithm is optimal because every node keeps informing one child in each step from the time the information became known to him, and it doesn’t matter which child because the subtrees are all the same.

**Exercise 2.2**

The worst case for starting broadcast is obviously a leaf. From there, the message needs \( m \) steps to travel to the root, and then only in step \( m + (k - 1) \), the last of the root’s children is informed. This is \( m - 1 \) steps later than in the above algorithm, which makes the time needed \( mk + m - 1 \).

**Exercise 3**

The only problem arising here is that a node cannot just send its computed broadcasting time to the parent as soon as it knows about it: If more than one child sends at the same time, the telegraph mode is violated and we don’t know what happens. To avoid this, assume that every node \( v_i \) in the tree has one token \( t_i \), which it can give to one child at a time to allow it to send messages up in the tree. Now, do the usual downcast-upcast-algorithm, starting from the root:

Every node that receives a message and is not a leaf informs its children, one by one, and after that sends its token to only the first child. Every leaf that receives a message sets its broadcast time to zero. On receiving back its own token, a node will pass it on to the next of its children which didn’t yet have it. If there is no such child left, the node will compute its broadcasting time in the way described in the lecture, and wait for a token to send it up to its own parent.

It is quite obvious that in this algorithm, each node will be receiving messages from at most one other node in each step: During the downcast, no child has the token yet and only the parent will send messages to a node \( v \). During the upcast, the downcast has already passed \( v \), so the parent will send nothing more, and only one child can because only one has the token.

We have a look at the running time of this algorithm. Due to exercise 2, the downcast-part will definitely run in time \( O((\Delta - 1)\text{depth}(T)) \). For the upcast part, note that the vertices on level \( \text{depth}(T) \) of the tree all know their broadcasting times right after the downcast because they are all leaves. Now analyze how long it can take until all nodes on level \( \text{depth}(T) - 1 \) know their broadcasting time.

For this, they have to wait for their at most \( \delta - 1 \) children to receive the token and send it back with their information. This takes time at most \( 2(\Delta - 1) \). Similarly, the next level \( \text{depth}(T) - 2 \) will consist only of informed nodes after at most \( 2(\Delta - 1) \) more steps, and so on. Therefore, also the upcast runs in time \( O(\text{depth}(T)(\Delta - 1)) \). (The root will need one additional step because it can have \( \Delta \) children, but this doesn’t matter in the Landau-Notation.)

To give a short argument for the policy used in the lecture to compute broadcasting times: It is clear that subtrees which take long should be informed earlier. It is also clear that if some subtrees take the same time, it doesn’t matter which to inform first. The simple observation that the overall broadcasting time equals the maximum over the times any subtree was informed, together with its own broadcasting time, directly leads to our policy.