Exercise 1.1

(I)

The complexity of echo algorithm is O(n)=2n because there are 2 **loops (line 29, and line 45)** that are iterating on the number of interfaces to send messages.

The first loop sends messages to its interfaces (so iterates on the number of interfaces it has) which triggers the nodes to send messages to their childs (child as the interface where the message is not coming from).

Those two-loop complexities added to each other is then equal to the number **n** of interfaces.

The loop line 45 will then not be used again as it is in an IF condition which checks if the node has sent a message (if color is not red, and line 49 in the IF sets the color to red preventing any more read of this part of the code).

Each message sent results to a count up of **rec**, which once equal to the number of interfaces the node has (and equal to the number of messages it has sent **minus** the one it received first) and sends a message to the father (the interface it has received the first message from). Each time the loop sends a message leads then to a second message sent back until it reaches the initiator.

**The time complexity is then loop one in initiate()+ the one in receive() = n**

**And e**ach time a message is sent, it is received and either leads to sending messages to the childs which is counted in the complexity above, nothing if the count for messages received is not equal to the number of interfaces equal to a complexity of 1, or a send back to the father which is done once on each interface (as the number of messages sent has to equal the number of interfaces once) so a complexity of n

**Therefore, the complexity is then 1+n+n = O(n) = 2n**

The message complexity is n for n messages sent to the childs and n for n messages sent back so 2n in total.

(ii)

A node knows only the number of messages it is supposed to receive (as shown in the question, only **rec** tells when to stop listening and send a confirmation message.

If a message is duplicated, the node will count it as a normal message and stop listening before receiving all the messages it should receive.

To avoid that, each node must listen to the first message of an interface only and ignore all the others.