Build it Up and Break it Down Activity (30-40 minutes)-Zero Robotics

1. During the phase of the program where basic programming theories are being introduced (You Have Nothing to SPHERE, Getting into CS, etc.), students gain practice understanding why and how their program needs to be written in a detailed and organized manner, so that it works properly when it is tested.
2. Split the class into two even groups (4 if there's a large class) with ideally 4-6 kids in each group. Give each group a box (or kit) of parts (Lego's, K'Nex, Lincoln Logs, blocks, etc.) that have exactly the same parts (color, shape, size, etc.) and quantities of such between them.
3. Have the groups work in separate rooms if possible, or on opposite sides of the classroom. If they're in the same room, they'll need to work a bit more quietly and discretely. Each group's task is to build a model (using ONLY those parts in their kits) that the other team will be able to replicate. The students should be told that they have time limits for what they're doing, but that they should be creative and build something unique and interesting that the other team can recognize and identify. The model shouldn't be too complex due to the time constraints.
4. Students should be given 10-15 minutes to decide on and build their models, plus write out detailed, step-by-step instructions for the other team to follow and be able to successfully duplicate their model that looks EXACTLY the same as the original model that was built. Remind teams that success in this exercise relies on if the *other* team can successfully follow their team's instructions and build the model they described correctly.
5. In the first part, students need to decide what their model should be (it should be something recognizable/familiar, examples include a house, a logo, a car, a geometric shape, etc.). Once they've decided, the team needs to build that model and, along the way, write detailed, organized, step-by-step instructions for the other team to follow so they can then build their model too. Sizes, shapes, and colors are all relevant elements that teams need to include in their descriptions.
6. Once the models are done and instructions completed, take pictures of the completed models so it can be verified, at the end of the 2nd part, that teams successfully followed the instructions and built the other team's model correctly. Have teams then switch places.
7. Within the time limit, the teams' jobs now are to read the other teams' directions and successfully build the model they intended to be built. Again, success depends on the accuracy of the sizes, shapes, and colors of the original models vs. what the other team builds.
8. After the time limit is up, have teams compare the pictures that were taken to the models that were built. Did the teams get it right? Were the instructions provided adequate?
9. Have teams discuss what happened. *Sample Questions:* How much detail was needed to get the builds 100% correct? Were the instructions the teams wrote adequate? Was more detail needed? What kinds of details (if any) were left out? What assumptions did the students make while writing their instructions that the other teams may not have made? How did this affect their final results?
10. Now, ask the students how these ideas translate to coding. *Sample Questions:* What assumptions or "initial conditions" are set for them in the IDE? What variables are used across the board, if any? How does the level of detail they use affect what their code ultimately does and how their device performs?