1. Please explain Big-O notation in simple terms.

Big-O notation provides a way to describe the performance of an algorithm separate from processor speed. As input size increases how is efficiency of the algorithm affected. It's a mathematical way to describe how fast an algorithm is. The lower the Big-O notation, the faster it is. Big-O notation allows you to compare algorithms and select the one that is the most efficient.

2. What are the most important things to look for when reviewing another team member's code?

The overall goal of reviewing code is to ensure that it is of as high quality as possible, that it functions as required without any major issues and that it can be easily maintained by others. Specifically these include:

- Use Cases – Does the code meet the stated requirements. Are there unaddressed bugs. Are there proper and meaningful outputs and error messages.
- Performance – Are the algorithms used the most efficient possible while still being understandable and clear. Is there an elegant structure and reuse of modules.
- Readability – Are the variable names descriptive and clear. Do the class and function names adequately describe their purpose. Are there enough comments which explain program flow.
- Conventions – Is the code written in the studio's agreed style and conventions.
- Unit Tests – Do enough tests exist which cover the essential functions and address common edge cases and errors
- Documentation – Are there clear instructions for tool usage. Is it properly formatted to ingest by the studio documentation system

3. Describe a recent interaction with someone who was non-technical. What did you need to communicate and how did you do it?

As an Animation Pipeline/Tool Programmer I often needed to address issues that Animators experienced. In one typical example, a very talented but non-technical animator contacted me about a problem he was having with an IK/FK switching tool he was trying to use for a character rig in his scene.

This particular tool was one that I had written and provided for a simple way to convert IK to FK animation curves and vice versa. If there was a problem with a particular rig, I had made sure that a clear error message was displayed which identified the problem and outlined who should be contacted in order to fix the issue. The chain of communication stated that the Animation Supervisor should be notified who would in turn contact the Rigging Supervisor for that particular character rig. However, since my desk was in the same room as the animators (which I loved, by the way), the animator knew it was my tool and asked me for help directly.

He called me over to his desk and as I approached, I could already tell he was frustrated that he could not do what he wanted in his scene. I wanted to make sure that I diffused his frustration and wanted to provide him with a clear way to resolve the issue. I made sure I did not remain standing over him but pulled up a chair so I was at his level. I invited him to show me the steps which led to the problem he was having. When the problem showed up again, I could already tell that this was a very common issue with newly created rigs. I assured him that he was doing everything correctly and we'd be able to address the issue. I asked him if this was new rig and he verified that it was and that he was doing tests before actual shot production began.

I asked him if he had noticed the error message that was displayed which said to contact his Animation Supervisor. He acknowledged that he had seen it but since I was so readily available he decided to call me over to make sure. I responded that I was always ready and willing to answer questions and thanked him for contacting me. He wanted to know why this rig did not work with the tool but all the other rigs on his previous show did not have any issues. This potentially needed a very technical answer which got to the very heart of how this tool worked in the character pipeline.

I explained to him that the IK/FK tool worked with all rigs as long as they had been set up correctly by the riggers. The tool was designed to be rig agnostic so that animators did not have to know anything about how the rig had been constructed. It just worked. I asked him if he realized that limbs on rigs could be constructed multiple different ways, or there could be different naming conventions across different rigs, or actually there could be third party rigs from other sources. The tool needed to work consistently in all those situations. I explained that there needed to be consistent tags added to the various controllers and joints to describe their particular role. This data needed to be added by the riggers and sometimes they had so much to do when a rig was first created, that this step could be overlooked. We just needed to let them know that this step was needed at this point for him to complete his task.

He expressed now that he understood what was needed, he would move on to another task and would contact his supervisor to get the priorities juggled. I noticed that after this, he volunteered this information to other animators when they had similar problems.