**1) What project have you been most proud of participating in and why?**

The project that immediately comes to mind with this question is my work at HoloLens. Being in the forefront of new and exciting technology like this device opened my eyes to what future might hold. In this QA team, we had to be the guardians of each OS build and giving it the green light when we felt it met our standards. Many times, we were in charge in communicating with several different teams in charge of HoloLens both in reporting issues as well providing assistance using the device effectively.

In this project, the responsibilities were vast as well as rewarding. It was a team that promoted individual effort which led to improvements to the team. One such case was during a time when the project was slowing down, I analyzed our Unity based testing tool and wrote up a proposal to convert completely manual test case into a hybrid of a manual test case with some automation for a case that required human eyes to check. It was approved and is still being used today, to my knowledge.

My work in HoloLens did a few things that I carry with me to this day, it let me discover my interest in the AR/VR space, it let me take initiatives and ownership that had not been possible in previous projects, and let me use Unity in a professional setting, and allowed me to positively affect the development of a successful device that is still being used till this day and is being used for the development of the next iteration. Both in the process and the result, HoloLens was a joy to work for.

**2) In a log(x) function, what are the boundary conditions?**

X > 0, x = infinity

**3) You are given a function, f(x), which is storing a number, N, between 1 and 1024. When you call the function it will return:**

**a. -1 if x<N**

**b. 1 if x>N**

**c. 0 if x == N**

**What is the minimum number of guesses you need if you want the function, f, to return 0?**

9

**4) How many bugs can you release a product with?**

In theory, the amount of bugs any product should be released with is zero. However in practice, the number of bugs a product can be released with depends on the type of product in question, time constraints, and resources.

Experience has shown me as QA, you try to plan ahead to catch most bugs, especially critical pri 0s such as bugs that freeze the app/system, corrupts visually, corrupts memory, and/or exposes security risks. Any product should never knowingly be released with any of these types of bugs. Other lower priority bugs may be present at released, however as QA, you strive to communicate to the product manager what these are and the impact on the app so they can make an informed decision.

The nature of some products can be more or less prone to bugs. Apps like games are harder to completely zero out due to how visual and unpredictable the user can interact with the game. But other apps are much more predictable and can be released with nearly zero bugs. To mitigate either type, automation testing the golden path of an app is critical. This should always test for known cases and scenarios to prevent the release of these bugs.

Time constraints on an app alongside with resources sometimes can cause compromise on how many bugs are present at release. The best thing to prevent this situation is to use automation on pri 0s, communicate with upper management, document the known issues to be worked post release, and to always strive for zero bugs.

**5) What is the difference between an interface and a delegate?**

While both can be very similar, delegates can only have one method; interfaces can have several methods as well as properties. Delegates can be implemented several times, while interfaces can only be implemented once.

**What do they have in common?**

They both share similar in that they are called like methods and created in a different object.

**When would you use one over the other?**

One clear distinction is how delegates are supported by the framework which allows for multicasting and asynchronous invocations. A good instance to use a delegate is with a LINQ expression where more than one implementation of the method may be needed.

**6) Why do you want to work for Unity?**

Unity is an engine that has interested me since my time at DigiPen. It really does a great job of balancing ease of use for beginners with complexity that can allow professionals to make amazing games that become highly successful. One thing I enjoy the most is to try to get more people interested in technology. Whenever I recommend a language to learn to program, I normally tell them to try C# so they can then use Unity. This has allowed me to get people who otherwise might give coding a chance, a way to see what is possible. Unity’s ease of use and availability has helped me turn those people into people who later pursue coding on their own.

Knowing that such a tool is not only a great start for beginners, but an industry standard used in gaming as well becoming a standard in many other industries such as Lowe’s, gives me great confidence in the future and vision of the Unity engine and company as a whole.

What better way than to contribute into such an amazing and inspirational tool than to be in the frontlines once again, similar to my experience at HoloLens.

# Assignment 1: Car Building

## Do you spot any obvious problems?

The numbers specified in the diagram do not match the specs provided needed for a car. For example it says it needs exactly one carburetor but the diagram allows for zero. It also allows for zero wheels when the specs state that it needs one or more.

Also the diagram linkage seems a bit off. Due to the function of carburetors, it would be more prudent to have them linked to the engine rather than directly to the car itself.

How does the design react to change?  
It easily allows for multiple kinds of each component, i.e. wheels, engines, and carburetors. However, it doesn’t scale well when it comes to other possible layouts or for future kinds of components such as an electric motor.  
Can we unit test it?

Absolutely, each component can be tested against the specs provided from the start. This will allow for unit testing in the car itself more efficient.