1. Software, in short, is the programs and processes designed to be run by a computer for various reasons. Video games, operating systems, social media apps (such as Facebook), and smartphone apps are just a few examples of different kinds of software.
2. The Seven Types of Software Applications:
   1. System Software: system software is programmed with the sole purpose of running a computer’s hardware; operating systems, such as Windows, MacOS, and Linux, are all examples of system software. System software also spies on you and sells your information for advertisement purposes.
   2. Application Software: application software is designed to complete specific tasks, such as word processors, video and photo editors, and calculators. Application software also spies on you and sells your information for advertisement purposes.
   3. Engineering/Scientific Software: this kind of software is made to help conduct theoretical experiments and enact real-world scenarios to help test hypotheses that would be much more difficult otherwise.
   4. Embedded Software: embedded software is a kind of software that cannot be found when navigating a computer in the typically manner. Embedded software is actually designed for the specific pieces of hardware inside computers, and is what allows them to perform the tasks they were designed for.
   5. Product-line Software: software product-lines is a kind of software development that refers to the methods in which said development takes place. This production technique allows for similar software systems to be created using similar assets, which creates a compatible collection.
   6. WebApps (Web Applications): web applications are pieces of software stored on a remote server that can be retrieved via internet within a browser. Web Application software also spies on you and sells your information for advertisement purposes.
   7. A.I. Software: artificial intelligence, in short, are programs designed to be “self taught.” These programs can simulate thousands of scenarios within seconds, and learn from their outcome.
3. The quote, “If hardware is the body of an organism, then software is the life,” is a simple yet relatively accurate way to describe how a computer functions at its core. In a living organism, the physical is what allows the mental/spiritual to exist. The same can apply when referring to the hardware and software of a computer; a piece of software cannot be run if there is no hardware to run it.
4. The quote, “Once a software is built for a customer and launched, our duties as a software engineer are done and we can move on to other projects” is not something I personally agree with. In my opinion, software can very rarely be considered “completed.” As a matter of fact, I believe that the real hard work begins once the software is released, as users will begin to submit bug reports and request features that a development team needs to take into account. I believe a good development team will focus on first refining newly released software before said focus can be shifted onto future projects.
5. The Attribute of Good Software:
   1. Maintainability: good software must be adaptable to change; consumers are expected to want changes made as time goes on, and the lifeline of software is extremely important. Well-designed software will be able to survive extreme change.
   2. Dependability/Security: software that is well-made will be able to assure the security and privacy of its users, and demonstrate that such assurance is followed-through. Furthermore, good software must be reliable. Customers must be able to trust that a product will work as advertised.
   3. Efficiency: something that must be considered when designing software is how it manages system resources, such as memory, storage space, and processing times. A lot of what defines good software lies in its efficiency, and is one of the most important aspects in my opinion.
   4. Accessibility/Acceptability: well-designed software must actually be acceptable by the target users. Compatibility across multiple kinds of operating systems and devices is crucial in building a reputation with a team’s consumer base.
6. Microsoft Windows, in my opinion, is a terrible operating system that follows none of the attributes of “good software.” Windows 11, the most recent release of Windows, is nothing more than a re-texture of Windows 10, and offers little-to-no improvement over its predecessors. Bugs are encountered at an alarmingly frequent rate, and the security that this operating system promises is nothing short of a lie. Compared to other mainstream alternatives, such as MacOS and Linux, the processing speeds of Windows is terrible. Furthermore, the OS at a whole is completely unreliable. Random blue screens of death are not uncommon, and system corruption is entirely possible for almost no reason. My frustration with Windows is endless, and goes against all the attributes of good software.
7. Software process models are a set of pathways that a development team would follow to create well-designed software, and can vary from project to project. Models are extremely important when it comes to developing software that is efficient and bug-free as possible.
8. I believe that a software-model is definitely a smart development idea, and a planned model should be created prior to creation. However, I also believe that the steps of said model are very much dependent on exterior factors, and can be tentative to change. The type of software being developed, the time window before scheduled release, and the size of the development team are all examples of these factors.
9. Student Attendance Management – The Prescriptive Model:
   1. Communication: I would ask the school what sort of features they were expecting to see in the program, such as information they wanted to be displayed for each student, the login requirements for staff access, and more.
   2. Planning: I’d work with my team to assign specific tasks to each member, as well as drawing out a map in which everyone can follow. I would make sure everyone was on the same page at all times, and that all the features requested by the school are planned to be implemented. Furthermore, I’d work with everyone to create a schedule to ensure that all members have the proper time to complete their assigned work.
   3. Modeling: I’d work with the team to develop a detailed model to focus on how the application is going to look visually, as well as how it will perform the required tasks.
   4. Construction: Once all the prerequisites have been completed, the team will then begin to work on the programming. Everyone will focus on their roles and collaborate together to make sure all features are compatible and functioning.
   5. Deployment: upon completion, the software will be sent to the school for testing and feedback. Any bugs will be corrected and fixed, and any features will be optimized upon request.
10. “Coding is the most important aspect of software development. Because without it, how will your software even exist?” This quote is not something I necessarily agree with. While acknowledging the fact that software would not exist without programmers, I believe that there are a multitude of other factors needed to be taken into account. Graphic design, for example, is critical in creating a software that attracts users. Most people are not software engineers, and therefore the general consumer will wind up favoring a program that is visually appealing. This is just one of many examples of factors that match up to the importance of coding.