**Question 2: It is said that "each minute spent planning/designing software will save many minutes in programming / debugging software." Why is this true?**

This is true because if you were to plan everything out before programming, you wouldn't need to think about your design while you code, which can lead to you needing to restart or remove certain features as you're working on the project. The later you find an error in your code, the more work and time you are going to have to spend to resolve the problem.

**Question 3: List the five major steps in the Waterfall SDLC and briefly explain the purpose of each step.**

**Analysis**: In this stage, the program gains a purpose through interviews or questionnaires to determine how people can use it and what features are needed. This stage will produce a requirement specification document containing the program's purpose.
**Design**: The design stage turns the requirements specification document into a design document, which will state how the program will be able to meet the requirements.
**Implementation**: During implementation, the programmer must convert all of their designs into code, and if the design is good quality, the implementation is usually very simple.
**Testing**: People must test the program and check if the program works as expected, but also find and report any additional bugs or errors they encounter.
**Maintenance**: This stage is after the program has been released, and users encounter and report bugs found in the program. The developer then performs maintenance on the program and fixes these errors after the program has already been released.

**Question 4: What is the greatest strength and weakness of the Waterfall Model?**

The greatest strength of the Waterfall Model is how easy it is to understand, regardless of your proficiency in programming or computers. However, the greatest weakness of the Waterfall Model is that since the water is flowing in one direction, it's really difficult to move back up the waterfall if you encounter a bug or error later down the road, as this could lead to a very painful and annoying time debugging your code.

**Question 5: How are pseudocode and flowcharts different?**

Pseudocode and flowcharts have a few key differences, as they are two different ways of representing algorithms. Pseudocode works by creating steps for how you will receive a result, normally written in English rather than in code. It's used as instructions for how to write your code, providing a rough idea on what you're going to do. On the other hand, flowcharts are much more graphical representations using shapes and arrows to represent the flow of actions your program must execute. While they are nice to help visualize what a program or algorithm does, they also take time to draw and may not be the most efficient use of time.