Part 1:

1. What is code review?
   1. Code review is a peer review process that is used to catch bugs, ensure code quality, and to ensure that best practices are followed, such as performance optimization and security checks.
2. Why is it an important practice for computer science professionals?
   1. Code review is important for the simple fact that it helps to improve the quality of your software, it will catch any potential issues earlier, and it will also enhance team collaboration.
3. What are some code review best practices that you read about in the resources that are crucial to include in a code review? Include when a code review should occur in the development process with a rationale as to why.
   1. The first best practice would be to use checklists. Using checklists ensures that everything on the checklist is completed. This will make sure that any previous steps are not skipped.
   2. Second, we need to ensure that we utilize constructive feedback. Using constructive feedback will help to improve the overall program.
   3. Finally, if it’s ever possible to automate a simple process, it would be best to do so. This will make those more simplistic processes quicker and they can be completed while members of the team are able to handle more complicated issues.
   4. The best time to perform a review is when a section of code has been completed, but BEFORE the new section of code is merged with the primary file. This ensures that the new section of code is sufficient before potentially harming the rest of the program.

Part 2:

1. What software have you chosen to use to record your code review?
   1. For my code reviews, I will be using GitHub, as its pull request feature will allow for adding comments, seeing version comparisons, and tracking progress.
2. Describe your approach to creating an outline or writing a script for your code review for each of the three categories that you will be reviewing based on the rubric as well as the code review checklist.
   1. To ensure a thorough code review, I’ll focus on three key areas: Structure, Loops and Branches, and Defensive Programming. First, I’ll check that the code accurately implements the design, follows coding standards, and avoids unnecessary complexity or repetition. For loops and branches, I’ll make sure conditions are properly nested, common cases are prioritized, and termination conditions are clear and achievable. Finally, in defensive programming, I’ll verify bounds checking for indexes and pointers, validate inputs, and ensure resources are managed correctly, such as deallocating memory and handling file or device errors. This approach helps keep the code efficient, reliable, and maintainable.