Area of evaluation	1- Fail	2 - Fair	3 - Good	4 - Outstanding
Code Behaviour (code compilation, execution)	Code doesn't run/compile due to issues with the code itself or with instructions provided; Code runs/compiles but numerous obvious bugs exist which may result in unexpected behavior.	Code runs/compiles but logic errors exist for basic edge cases which may result in unexpected behavior.	Code runs/compiles, considers basic edge cases, may have debatable logic or not consider subtle edge cases.	Code runs/compiles, considers all edge cases. Thorough error handling. Bonus if it includes informative logs.
Solution Design (logical separation, concepts abstraction, OOD, code/folder structure)	No attempt for OOD programming with separation of concerns in mind, or an attempt was made but was incorrect or incomplete.	Depending on the assignment, the code follows OOD programming, such as implementing a class rather than a function. Basic logical concepts were properly implemented based on separation of concerns. Code was organized in separate files.	Depending on the assignment, the code follows OOD programming, such as implementing a class rather than a function. ***All*** logical concepts were properly implemented based on separation of concerns. Code was organized in separate files/folders.	Considers OOD programming depending on the assignment. All logical concepts were properly implemented based on separation of concerns. It went above and beyond to make efforts to address extensibility, scalability, or future use cases. Code was organized based on the best practices for the language.
Code Testing (code coverage, edge case consideration)	No unit tests, or they fail without any explanation.	Unit tests exist and pass, include tests for basic happy paths, but may not cover edge cases or all code.	Unit tests exist and pass, include tests for happy paths as well as some edge cases. Cover all code.	Unit tests exist and pass, include tests for all happy paths as well as all edge cases. Cover all code.
Readability (self-explanatory code naming, comments, documentation, README)	Code is unreadable, or confusing to understand.	Code is generally readable. Code names can be improved. May lack comments to clarify data structures or technical decisions.	Code is pretty readable. Code names are self-explanatory. Sufficient comments to clarify data structures or technical decisions. Basic README/documentation for environment setup or how to run the code etc.	Code is very readable. Code is self-explanatory. Code is written based on the best practice of the language. Sufficient comments to clarify data structures or technical decisions. Detailed README/documentation for environment setup or how to run the code etc.