Criterion (based on ACA Unpacking elements)	***A Standard*** *85-100% of marks available*	***B Standard*** *70-85% of marks available*	***C Standard*** *55-70% of marks available*	***D Standard*** *40-55% of marks available*	***E Standard*** *0-40% of marks available*
Data representation (Binary + Text) /10	The solution to level 7 correctly uses binary to represent unicode, and binary representation and operations (e.g. bit-level logic) are used in extension levels.	The solution to level 7 correctly uses binary to represent a unicode character, and binary representation is also used in extension levels.	The solution to level 7 uses binary encoding as part of the question or answer. The binary encoding should represent a unicode character(s) (such as an emoji 👍).	The solution to level 7 attempts to use binary in the question or answer, but the implementation is incorrect.	There is no attempt to use binary in the question or answer in level 7.
Specification /5	Meets all of the DLC/extension requirements as specified in the assignment.	Meets some of the DLC/extension requirements as specified in the assignment.	Meets all requirements as specified in the core part of the assignment.	Meets the core requirements of the assignment, but some are incomplete or partially implemented.	Very few of the core requirements are implemented, or all of them are incomplete.
Implement Digital Solutions + General-purpose programming /20	Variables used track the state of the program, are named well, hint about data types and are concise. Comments are descriptive, the code consistently follows style conventions and is modular/split up across	Variables used track the state of the program, have names that help the reader understand and follow the code, and provide hints about the data type. Comments describe key functionality, specify arguments, and the code consistently	Variables used track the state of the program and have names that help the reader understand and follow code. Comments describe the purpose of each level/function and any arguments required	Variables used in the program have ambiguous names. Comments are present but are either incomplete or unnecessarily detailed.	Variables used in the program are either non-descript or confusing/named inappropriately. Comments are either missing or do not describe the purpose/operation of the code.

	a sensible file/folder structure.	follows appropriate style conventions.			
Test Digital Solutions /15	The student submits tests for all levels in the game, and presents inputs used and output generated. All extension levels are included. The tests used include expected and unexpected values, edge cases, and demonstrates a range of different ways unexpected input is handled.	The student submits tests for all levels in the game, and presents inputs used and output generated. The tests used include expected values and some unexpected values that users may enter, and demonstrates the “default” behaviour with incorrect results.	The student submits tests for most of the levels in the game, presenting the inputs used and resultant output. The tests used include values expected as input from the player, and the program output produces correct results.	The student submits tests for most of the levels in the game, but some do not include a complete set of inputs. The tests submitted do not consistently produce correct results.	The student either submits no tests, an incomplete test suite, or tests that yield a lot of incorrect results.
User input /10	Input handles different data types including text, numbers and emoji. Input correctly manages case - with appropriate output when there is an error. Prompts are at correct locations and are clear and succinct. Prompts guide the user to complete the program. Provides additional commands at any point in the game.	Input handles different data types including text, numbers and emoji. Input correctly manages case. Prompts are at correct locations and clear and succinct.	Input handles different data types including text, numbers and emoji. Prompts are at correct locations and are clear.	Input handles data types correctly but does not handle other types of data, for example numbers or emoji do not function correctly. Prompts are usually placed well and are not always clear.	The program does not process the input correctly. Prompts are incorrect or missing.

Control structures (branching and iteration) /20	Control structures ensure program flow is correct and use nesting appropriately. A range of comparison operators are used, logic is simple and easy to follow, and is easily extensible for new features to be incorporated.	Control structures ensure program flow is correct and use nesting appropriately. A range of comparison operators are used, and logic is kept simple to ensure readability.	Control structures ensure program flow is correct and use nesting appropriately. A range of comparison operators are used.	Control structures ensure program flow is correct, but may have minor bugs. Comparison operations used are limited to equal / not equal comparisons.	Control structures are haphazard and don't always result in the correct program flow. If they are used at all, comparison operators are limited in scope.
Functions /20	Individual functions are used with parameters and arguments, return values, and helper functions are used throughout the code to implement player interaction effectively.	Individual functions are used for each level, parameters and arguments used correctly, and return values to the main program to update and manage the program's state.	Individual functions are used for each level, required parameters are defined, and necessary arguments passed to each function when called.	Functions are used for level, but some errors or inconsistencies are present either in the definitions themslebes or when called in the main program.	Functions are either not used, or used incorrectly throughout the program.