Unit:
Introduction to Programming with Python

Assignment title:
Snakes and ladders

[Cycle] [Year]

Marking Scheme

Markers are advised that, unless a task specifies that an answer be provided in a particular form, then an answer that is correct (factually or in practical terms) **must** be given the available marks. If there is doubt as to the correctness of an answer, the relevant NCC Education materials should be the first authority.

This marking scheme has been prepared as a **guide only** to markers and there will frequently be many alternative responses which will provide a valid answer.

Each candidate's script must be fully annotated with the marker's comments (where applicable) and the marks allocated for each part of the tasks.

Throughout the marking, please credit any valid alternative point.

Where markers award half marks in any part of a task, they should ensure that the total mark recorded for the task is rounded up to a whole mark.

Marker's comments:

Moderator's comments:

Mark:	**Moderated mark:**	**Final mark:**

Penalties applied for academic malpractice:

Design

	Guide	Maximum Marks
1	***Success criteria*** • ***5 marks: Comprehensive set of success criteria that cover all aspects of the game and are measurable*** • ***4 marks: Success criteria that cover all of the game and the majority are measurable*** • ***3 marks: Success criteria cover most aspects of the game, they are mostly measurable*** • ***2 marks: Success criteria cover many aspects of the game and some are measurable and extend the brief*** • ***1 mark: Some success criteria are given but they repeat the information in the brief, they are not measurable*** • ***0 marks: No attempt at success criteria***	**5**
2	***Decomposition*** • ***5 marks: Appropriately decomposed problem that is logical, with range of modules including identification of repeated elements and functions*** • ***4 marks: Problem is decomposed to several levels, some modules identified and functions. Some areas could be decomposed further.*** • ***3 marks: Decomposition has used, although all application may not be appropriate, some areas missed or still high-level*** • ***2 marks: Attempt at decomposition, including splitting the main problem down into several sub tasks, but these may not be split further. No repeated elements identified.*** • ***1 mark: Attempt to decompose, but at a high level and may only include one or two modules.*** • ***0 marks: No decomposition***	**5**
3	***Structure diagram*** • ***5 marks: Multiple structure diagrams (or one large) that covers all aspects of the game, in appropriate format and structure, with modules clearly identified*** • ***4 marks: Structure diagram(s) follow the correct format and structure, some elements may not be fully accurate or clear as to how they are connected*** • ***3 marks: Structure diagram is mostly in appropriate format, but some areas may have inappropriate connections. Some elements of the game are missing.*** • ***2 marks: Structure diagram of at least two levels that covers many aspects of the program.*** • ***1 mark: Attempt at structure diagram that includes at least one main problem and a sub problem.*** • ***0 marks: No structure diagram***	**5**

	Guide	Maximum Marks
4	***Pseudocode*** • ***Class design for board (1 mark) including appropriate constructor (1 mark), attributes (1 mark), methods (1 mark). Use of the class methods to access the board contents throughout the design (1 mark).*** • ***List (or other appropriate data storage) for the board contents within the class (or external if no class) (1 mark) with correct number of elements (1 mark)*** • ***Opening text file for positions (1 mark) reading in data (1 mark) splitting the data (1 mark) and storing correctly in the board (1 mark) and closing the file (1 mark)*** • ***Use of random number to roll two dice (1 mark)*** • ***Alternative two players (1 mark) with appropriate messages and inputs for moving (1 mark)*** • ***Method/function to move a character's piece (1 mark) that calculates new position (1 mark) and changes the position of the piece (1 mark)*** • ***Checking if they position is a jump or go back (1 mark) and acting accordingly (1 mark)*** • ***Outputting the board states (1 mark) in an appropriate format (1 mark) that shows both player positions (1 mark)*** • ***Checking if a player is on the last position (1 mark) and outputting they have won (1 mark)***	**25**
	Total available marks for design	**40**

Implementation

	Guide	Maximum Marks
1	***Creating the program:*** • ***Class for board (1 mark) with constructor (1 mark) and list for the board contents (or other suitable method of doing this e.g. a second class) (1 mark)*** • ***Instance of the class is created and stored (1 mark)*** • ***Class has suitable get methods (1 mark) and set methods (1 mark)*** • ***Board has correct number of spaces (1 mark)*** • ***Text file is opened (1 mark) looped until EOF (1 mark) and all data read (1 mark) (or other method, e.g. reading all into a list and then iterating through) and file is closed in an appropriate place (1 mark).*** • ***Exception handling used to catch invalid/not found file (1 mark) with suitable output (1 mark)*** • ***Data from text file is split by commas (1 mark), board piece is extracted (1 mark) and jump/go back is stored in correct position (1 mark)*** • ***Random library imported (1 mark) two dice use random numbers between 1 and 6 (1 mark)*** • ***Players start in position 1 (1 mark) and roll the dice in turn (1 mark)*** • ***Roll number is added to current position (1 mark) piece is moved (1 mark)*** • ***New position is checked for jump or go back (1 mark) and actioned correctly (1 mark)*** • ***Play repeats between the two players (1 mark) until one player reaches the last space (1 mark) when an appropriate output is given (1 mark)*** • ***Board positions/content is output (1 mark) in appropriate places in the game (1 mark) in a suitable format i.e. the grid (1 mark)***	
	Total available marks for implementation	**30**

Testing

	Guide	Maximum Marks

1	***Testing strategy*** • ***15 marks: Testing strategy coves all elements of the game. Includes normal, extreme and invalid data for all areas. Includes tests with different text files. Includes testing game features and run-throughs of the game.*** • ***14 marks: Testing strategy coves all elements of the game. Includes normal, extreme and invalid data for most areas, some areas may have potential for further testing. Includes tests with different text files. Includes testing game features and run-throughs of the game.*** • ***13 marks: Testing strategy coves all elements of the game. Includes normal, extreme and invalid data for many of the areas, but may be limited in some for example only testing one type of invalid data. Includes tests with at least two different text files. Includes testing game features and run-throughs of the game.*** • ***12 marks: Testing strategy covers the majority of the game and includes clear normal, extreme and invalid data for most of these elements. The reading from the text file is included in the testing and the strategy includes run-throughs although these may be limited.*** • ***11 marks: Testing strategy covers majority of the game inputs and outputs, may miss some areas, for example run-throughs of the whole system. Normal, extreme and invalid data are used consistently.*** • ***10 marks: Testing strategy covers most of the game, although some areas have limited testing or are not included. Normal, extreme and invalid data are used in the strategy. The text file reading is included in the testing.*** • ***9 marks: Testing strategy covers most of the game, but some elements are clearly missing and others may have limited testing. Range of test data used, but inconsistently.*** • ***8 marks: Testing strategy covers many aspects of the game, these may only be tested once and not with repeated data.*** • ***7 marks: Testing strategy covers many aspects of the game, but some significant elements are missing. Testing may not include normal, extreme and invalid data.*** • ***6 marks: Testing strategy covers some of the game, but not all elements are tested. There is an attempt to use a range of data, but this may be limited.*** • ***5 marks: Testing strategy covers some of the game, but significant areas are missing. There may be a range of data used but not clearly identified.*** • ***4 marks: Testing strategy is suitably structured and covers some areas of the system, with an attempt to identify a range of data.*** • ***3 marks: Testing strategy has some structure, identifies how aspects of the game will be tested and the data it will be tested with.***	**15**

	• ***2 marks: Testing strategy is not structured appropriately, for example a description of how it will be tested, but there is evidence of how some aspects of the game will be tested.*** • ***1 mark: Testing strategy is generic statement of what will be done and not how it will be tested in a suitable format. Only some of the game will be identified.*** • ***0 mark: No testing for any of the system.***	
2	**Test log** • ***5 marks: Test log has evidence of the results of all tests alongside how they were tested. Evidence supports the tests.*** • ***4 marks: Test log has evidence of most of the results of the tests, some may be unclear as to how they are met.*** • ***3 marks: Test log has evidence for some of the tests but do not show that the system is functional.*** • ***2 marks: Test log has some evidence of tests, but these are limited and possibly unclear.*** • ***1 mark: Test log is completed with an attempt to show the evidence of at least one test.*** • ***0 mark: No test log produced.***	**5**
3	**Test against success criteria** • ***5 marks: Success criteria are systematically analysed with evidence produced or linked to show they have been met, or not met.*** • ***4 marks: System has been compared to success criteria and evidence for most is given or referenced.*** • ***3 marks: System has been compared to some of the success criteria, but there may be little evidence to support statements.*** • ***2 marks: System ha been evaluated but there may be no clear link to success criteria. Little or no evidence given.*** • ***1 mark: An attempt to evaluation the system is given, but without linking to success criteria and no evidence provided.*** ***0 mark: No testing for any of the system.***	**5**
	Total available marks for testing	**25**

Publishing programme and installation documentation

	Guide	Maximum Marks
1	Publishing programme and installation documentation • ***5 marks: Comprehensive documentation that shows how to install the program and use it. Documentation is appropriately structured and formatted and includes help or FAQs*** • ***4 marks: Documentation covers most areas of the program, but may not include some aspects for example how to use a different text file. Documentation is appropriately structured and easy to follow.*** • ***3 marks: Documentation only covers some aspects, for example does not show installation. Guide can be followed but there may be difficulties in the structure.*** • ***2 marks: Documentation is provided but is limited in content for example only shows basic instructions for playing the game. Structure and formatting is not easy to use.*** • ***1 mark: Attempt to create documentation that covers at least one aspect of the game, may be text-based and no images to illustrate use.*** ***0 mark: No testing for any of the system.***	**5**
	Total available marks for publishing programme and installation documentation	**5**

Note to markers

Please take appropriate action for any malpractice (plagiarism, collusion, referencing issues etc.) discovered as per the *AQ_28-a01_Academic Misconduct Policy* document. Please also complete and submit the *Malpractice Declaration Form*.

Learning Outcomes matrix

Task	Learning Outcomes assessed	Marker can differentiate between varying levels of achievement
Assignment task		Yes

Grade descriptors

Learning Outcome	Pass	Merit	Distinction
Describe and apply a systematic approach to the design of programs	Demonstrate ability to perform the task	Demonstrate ability to perform the task consistently well	Demonstrate ability to perform the task to the highest standard
Write small procedural programs to perform well-defined tasks, following well-defined requirements	Demonstrate ability to perform the task	Demonstrate ability to perform the task consistently well	Demonstrate ability to perform the task to the highest standard
Test and document program code following the principles of software engineering	Demonstrate ability to perform the task	Demonstrate ability to perform the task consistently well	Demonstrate ability to perform the task to the highest standard
Describe and apply the benefits of modular software design	Demonstrate ability to perform the task	Demonstrate ability to perform the task consistently well	Demonstrate ability to perform the task to the highest standard