OCR
Oxford Cambridge and RSA

Accredited

GCSE (9–1) Computer Science

J276/03 Programming project – Task 3

Sample Non-Exam Assessment

- Please check on **OCR Interchange** that you have the Non Examined Assessment material valid for the appropriate assessment series.

INSTRUCTIONS TO TEACHERS

- Please refer to Section **3f** of the GCSE (9–1) Computer Science specification for instructions on completing the Non-Exam Assessment tasks.
- The marking criteria should be available to candidates whilst completing the task.
- The quality of extended response will be assessed in the 'Development and Testing' and 'Evaluation and Conclusions' sections.
- Teachers are responsible for ensuring that the Non-Exam Assessment Material completed is valid for the appropriate assessment series.

INFORMATION FOR CANDIDATES

- The total number of marks for this component is **40**.
- This document consists of **4** pages.

Candidates should complete the task and provide evidence to meet all the marking criteria.

For the following scenario analyse the detailed requirements and using suitable algorithms, design a solution to be coded in a suitable high-level programming language.

Show the iterative development of the individual solutions with suitable testing throughout the process.

Test the final product and evaluate your solution against the detailed requirements you identified in the analysis.

The non-exam assessment must be done using a suitable high level language such as:

- Python
- C family of languages (for example C# C++ etc.)
- Java
- JavaScript
- Visual Basic/.Net
- PHP
- Delphi
- SQL
- BASH

You may use a combination of programming languages to produce a solution to the task.

Teachers **may:**

- explain the task
- advise on resources
- provide the support described within the 'Permitted Support' section of the Specification
- interrogate learners to ensure that the work is their own
- provide a copy of the mark scheme to candidates

Teachers **must not:**

- give detailed advice and suggestions as to how the work may be improved in order to meet the assessment criteria. This includes indicating errors or omissions and personally intervening to improve the presentation or content of the work.
- practise the task with the learners
- practise tasks which are similar in nature with the learners
- provide templates, model answers or feedback on drafts
- produce templates or model answers and publish them online.

Teachers **must** ensure that:

- learners do not access the internet*
- learners are not allowed to take the NEA tasks home with them.
- all work presented for submission must have been completed under supervised conditions.
- accounts associated with the NEA tasks must be locked between sessions to ensure that learners cannot access them outside of the supervised conditions.
- learners do not access online file storage accounts or email during the supervised conditions in order to prevent learners from completing work at home and bringing it into the supervised conditions.

*unless the centre is using an online IDE, in which case, only access to the IDE website is allowed

Scenario

Clark is designing a game that allows players to move around a 7x7 grid, each position in the board has a number to represent the space, as shown in Fig 1.

43	44	45	46	47	48	49
42	41	40	39	38	37	36
29	30	31	32	33	34	35
28	27	26	25	24	23	22
15	16	17	18	19	20	21
14	13	12	11	10	9	8
1	2	3	4	5	6	7

Fig 1

Players must move through the spaces in numerical order, starting from space 1, all the way to space 49.

Player(s) roll two 6-sided dice and move that number of places, e.g. if they are on space 4 and they roll a 3 and a 2, they move 5 spaces to space 9.

The winner is the first player to reach space 49. Players do not need to roll an exact number to reach space 49, for example if they are on 48 and roll 3, they will still win.

Analyse the requirements for this system and design, develop, test and evaluate a program that:

1. allows 2 players to play the game
2. allows the players to take it in turns to roll two 6 sided dice and move
3. display the result of each move on the board
4. makes a player move back the number of positions rolled if they roll a double (two dice with the same number)
5. displays the messages below when the condition for display is met (the condition is given to you below):
    1) Start Game message: displays either just before the first roll of the dice, or upon start-up
    2) A message when a ‘double’ is rolled (e.g. a 3 on each die): displays only when the score both die are identical
    3) Win message when they finish the game: displays when the player score is 49 or greater

    These messages should be:
    - stored externally and then read in to the game at the start of the program

Some games have obstacles or challenges that may send you back or forward by a set number of spaces.

6. Create a way of externally storing at least 4 “obstacles” and the number of squares they move forward or backward by.
7. Load these obstacles into the game when it starts.