Chhetri Academy

chhetriacademy.com

GCSE COMPUTER SCIENCE

6.1

Edexcel 1CP2 | Specification point 6.1.1

Be able to use decomposition and abstraction to analyse, understand and solve problems

Untiered

Answers and examiner comments

1	2023	Untiered
question groups	available papers	tier

Using this booklet

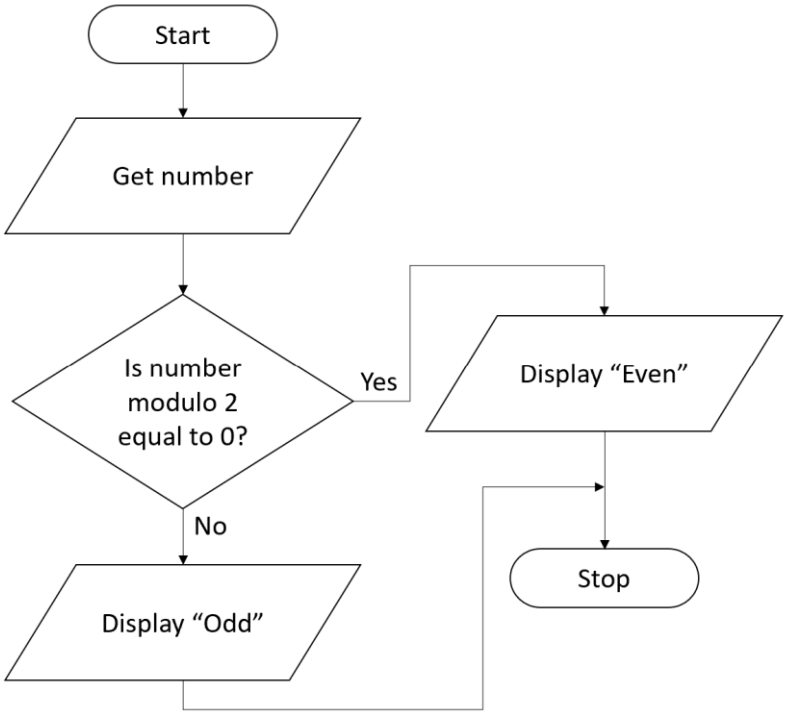
1. Answer numbers match the companion question booklet.
2. Follow the original question number on marking tables.
3. Examiner comments are included where available.

Name _____ **Date** _____

Some entries use full source marking material in an appendix. Any unavailable mark schemes are identified beside the relevant answer.

Question Number	Answer	Additional Guidance	Mark
4(a)	Award up to two marks for a linked explanation, such as: • It helps the programmer visualise the steps in a program / find errors/bugs / check the algorithm gives the correct output (1) because they can see the value a variable holds at a given point in an algorithm (1)	For both marks, the expansion must associate with the statement.	(2)

Question Number	Answer	Additional Guidance	Mark
4(b)	Award up to two marks for a linked description, such as: • A list / data structure / collection of elements/fields/values (1) where each element/field/value may be a different data type / is related to the others (1)	For both marks, the expansion must associate with the statement.	(2)

Question Number	Answer	Additional Guidance	Mark
4(c)	Award one mark for each of: • Input symbol to get number from user (1) • Decision symbol (disregard test inside), with exactly two arrows and labels for yes and no (1) • Correct relational test matches labels and output messages (1) • Accurately connected (1): o outputs 'Display Even' **or** 'Display Odd', not both o no hanging symbols o connection to terminator  	Arrows must match symbols for use of symbol to be awarded Each symbol should have only a single input arrow Input/output symbols should have only a single output arrow The decision symbol must have exactly two output arrows Relational tests: • X % 2 == 0 • X MOD 2 == 0 • X modulus 2 == 0 Allow = for == in relational test	(4)

Question Number	Answer	Additional Guidance	Mark
4(d)	Award up to four marks for a linked description, such as: • Start at the first position / Iterate/Traverse (through the array) (1), compare the item with the target (1), stop when the target is matched (1), or stop when the end of the list is reached (and the item is not matched) (1)		(4)

Question Number	Answer	Additional Guidance	Mark
4(e)	Award one mark for each of: • Four unique rows for the values of S and M, indicating binary representation of 0, 1, 2, and 3 (1) • Four correct results of applying AND to the values in S and M (1) Example:	Ignore order of rows Award both marks only if completely correct Award carry through in final column, if error in S or M columns Only award: - 1/T/True/On - 0/F/False/Off	(2)

S	M	S AND M
0	0	0
0	1	0
1	0	0
1	1	1

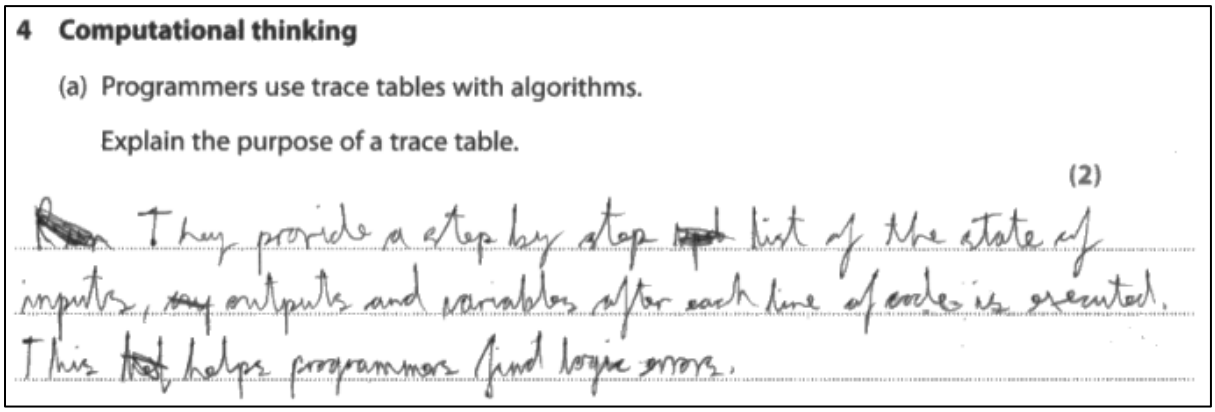
Question Number	Answer	Additional Guidance	Mark
4(f)	Indicative content: Definition • Decomposition is breaking down into smaller parts. Problems, solutions, and algorithms can be decomposed. • Abstraction is the process of removing or hiding unnecessary detail. Benefits • It is usually easier to solve several smaller problems, such as checking if touching a wall or updating the score display, than solve one big problem, such as making a game. • Different parts of the program can be shared between the class members to speed up development, for example, one group could work on the code to control the character, while another works on creating and playing the sounds. • Once all the pieces, like sounds, movement, and score are working correctly, the smaller solutions can be combined to make a larger solution, with fewer errors. • The individual parts of the program, such as updating the score can be ignored by the group of students writing the code for moving the character with the arrows / allowing each group of students to focus only on the small problem they have been given means time is not wasted on analysis not relevant to the solution. Appear in program code • Different blocks of code logic show decomposition, such as		(6)

	moving the character and updating the score. These blocks could be shown separated by white space. • Subprograms are decompositions because they're blocks of code logic, separated from the main program. Subprograms could be used for updating the score and resetting the character back to the starting position. • Abstraction is used each time a subprogram, either built-in, in a library, or in the code file is called. Library routines in the game would include one to play the sounds and to get the keyboard input. • Subprograms are abstractions because their names hide how they work internally, even if their name is not descriptive. A subprogram to update the score should have a descriptive name, such as updateScore, but could just be called X. Either way, how it works internally is still hidden from the caller.		

Level	Mark	Descriptor
	0	No rewardable content.
Level 1	1–2	Basic, independent points are made, showing elements of understanding of key concepts/principles of computer science. (AO1) The discussion will contain basic information with little linkage between points made or application to the context. (AO2)
Level 2	3–4	Demonstrates adequate understanding of key concepts/principles of computer science. (AO1) The discussion shows some linkages and lines of reasoning with some structure and application to the context. (AO2)
Level 3	5–6	Demonstrates comprehensive understanding of key concepts/principles of computer science to support the discussion being presented. (AO1) The discussion is well developed, with sustained lines of reasoning that are coherent and logically structured, and which clearly apply to the context. (AO2)

Question 4a

While some responses achieved both marks for a well-linked explanation, there were many that got trace tables confused with truth tables. Most correct responses mentioned that they are used to check the correct output, or test for bugs, but failed often to explain how, omitting the fact that a trace table tracks variables.



4 Computational thinking

(a) Programmers use trace tables with algorithms.

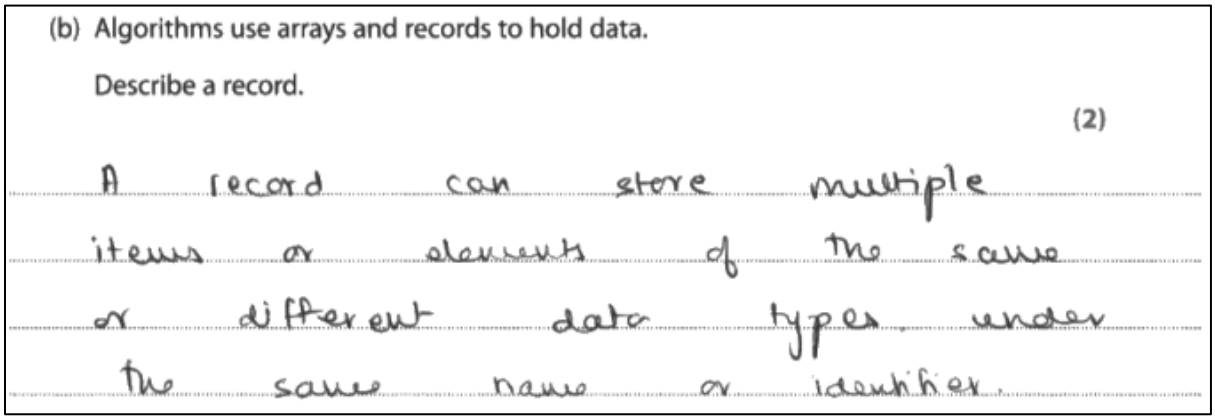
Explain the purpose of a trace table.

(2)

They provide a step by step list of the state of inputs, outputs and variables after each line of code is executed. This helps programmers find logic errors.

This response scored 2 marks. 'They provide …. the state of … variables' gained the first mark and ' helps programmers find logic errors' gained the second mark.

Lots of responses described storage or a storage method, rather than a data structure.


(b) Algorithms use arrays and records to hold data.

Describe a record.

(2)

A record can store multiple items or elements of the same or different data types, under the same name or identifier.

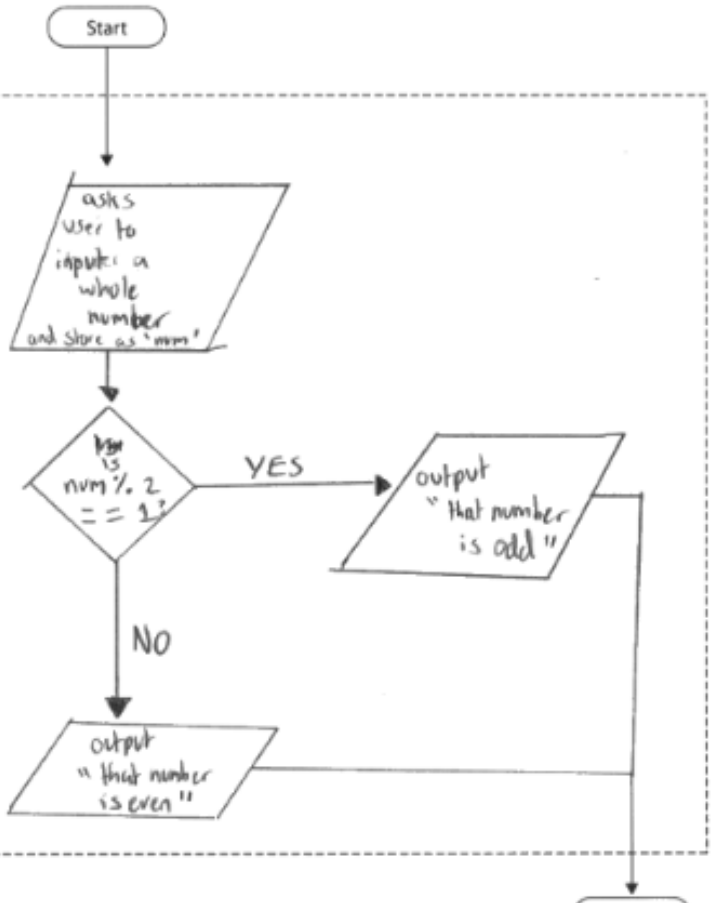
This response scored 2 marks; 1 mark for 'can store multiple items or elements' and 1 mark for 'different data types'.

Question 4c

Most responses correctly used the appropriate input symbol and had two outcomes from the decision. They understood the necessity of evaluating conditions and making choices based on them. Some students struggled to comprehend the relational test involving MOD, instead focusing on DIV, or whether the value divided by 2 had a remainder. Despite this, many students could make appropriate connections to the terminator and include at least one output message relevant to the decision outcome.
A few students used a decision shape instead of an output shape.

You may not need to use all the flowchart symbols.

(4)



This response scored all 4 marks.

Question 4d

Some responses methodically outlined the steps of the algorithm. However, the final mark point awarded for when the item is not found so the algorithm stops and the explicit point of comparing the item with the target were the most frequently missed marks. Many responses deviated from the intended focus by discussing sort algorithms.

(d) A linear search algorithm can be used on both a sorted and an unsorted array.

Describe how a linear search algorithm operates on an **unsorted** array.

(4)

It iterates on every value of an unsorted array and for every value checks if the value is equal to the value being searched for. If not, it will continue to the next variable until it does. If it reaches the end of the list and the value has not been found, it will declare that there is no such value

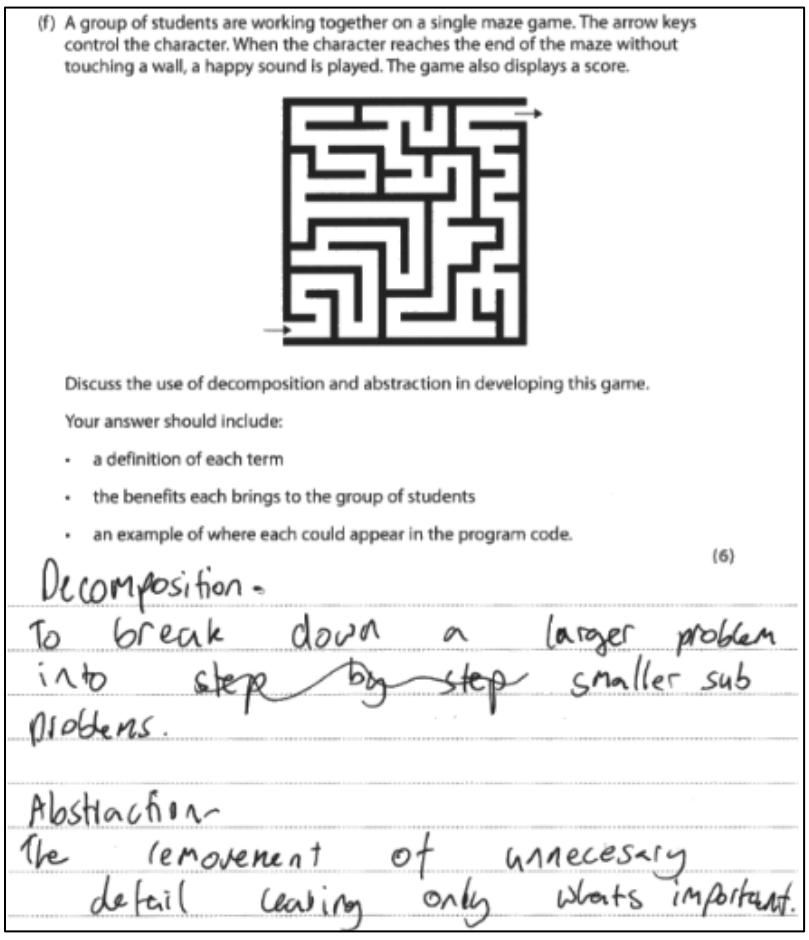
This response scored 4 marks. 'iterates' 1 mark, 'checks if the value is equal to the value being searched for' 1 mark, 'if not, it will continue until it [is equal to the value being searched for]' 1 mark and 'If it reaches the end of the list and the value has not been found' 1 mark.

Question 4e

A number of responses used ticks and crosses instead of 0, 1 etc. In general, students fully understood the logic process.

Question 4f

Despite being given bullet points to cover, many students did not use the structure of the question to compile their answer. Many cited vague benefits and although the definition for abstraction was often given, students were not clear on its benefits or its use. For example: some students thought abstraction would save on file size if a simplified character was used, while others talked about using or simplified maze, because they would be easier to navigate.



(f) A group of students are working together on a single maze game. The arrow keys control the character. When the character reaches the end of the maze without touching a wall, a happy sound is played. The game also displays a score.

Discuss the use of decomposition and abstraction in developing this game.

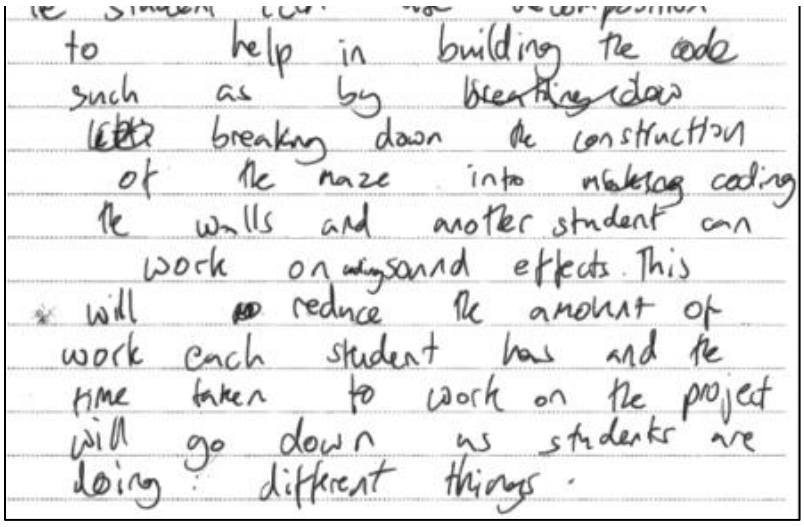
Your answer should include:

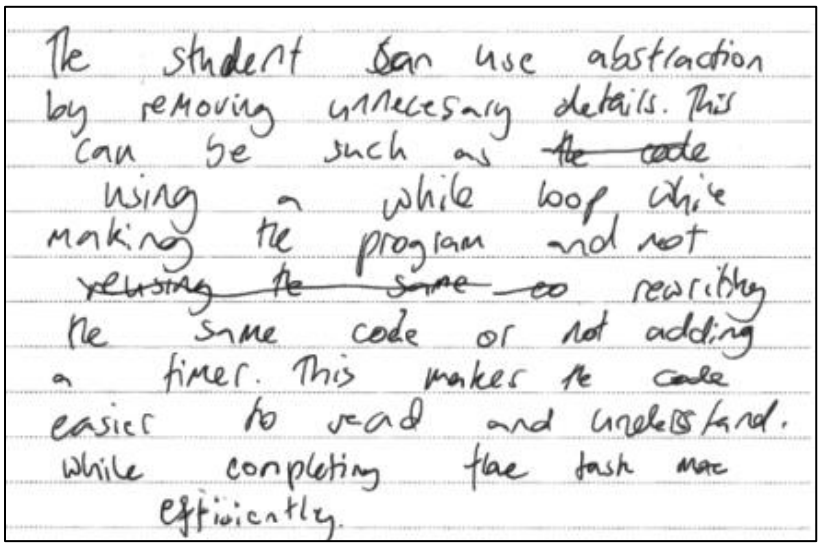
- a definition of each term
- the benefits each brings to the group of students
- an example of where each could appear in the program code.

(6)

Decomposition -
To break down a larger problem into step by step smaller sub problems.

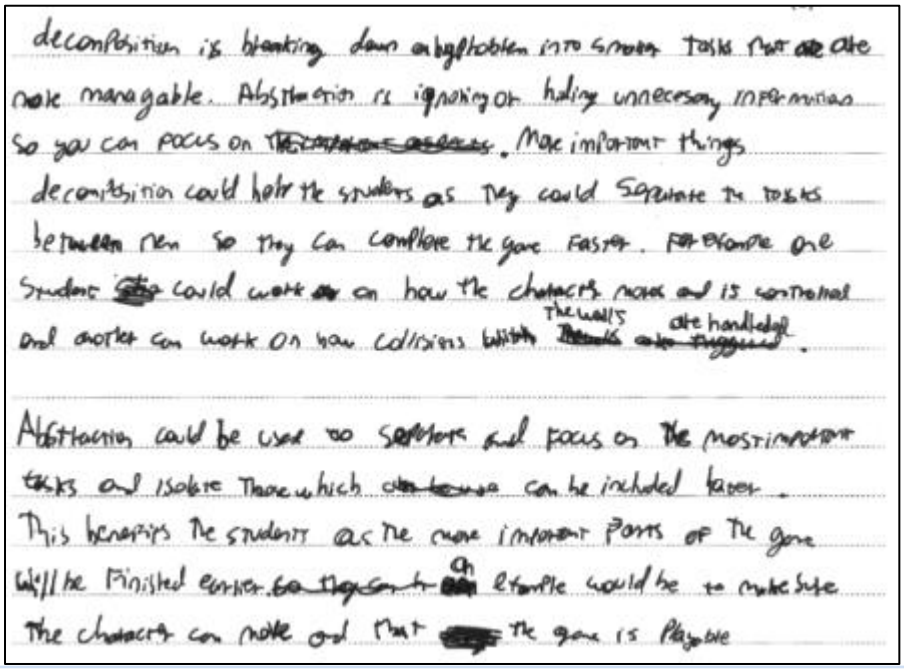
Abstraction -
The removement of unnecesary detail leaving only whats important.


to help in building the code such as by breaking down the construction of the maze into making coding the walls and another student can work on adding sound effects. This will reduce the amount of work each student has and the time taken to work on the project will go down as students are doing different things.

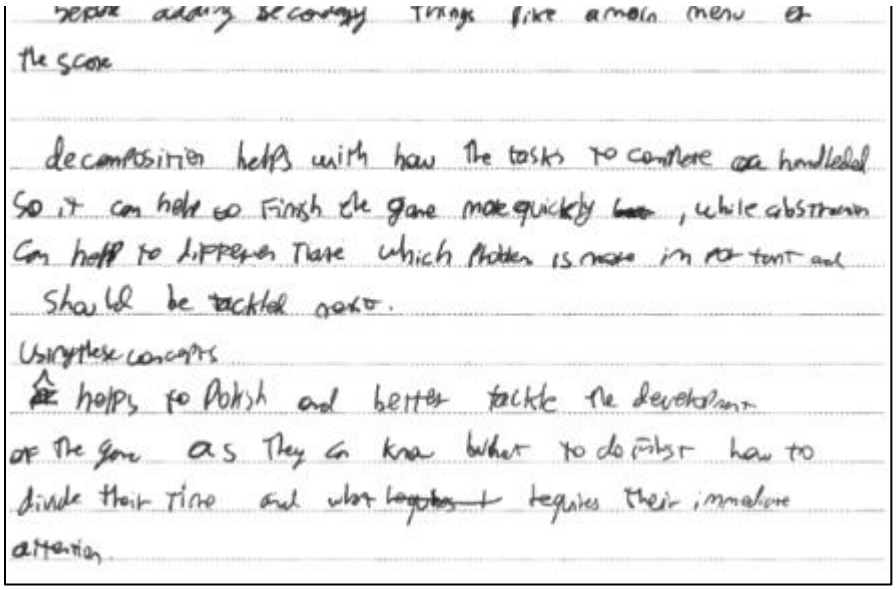

The student can use abstraction by removing unnecesary details. This can be such as the code using a while loop while making the program and not repeating the same code or not adding a timer. This makes the code easier to read and understand. While completing the task more efficiently.

This response scored 3 marks. Both aspects are defined, and links are made to the benefits of decomposition. The benefits of abstraction are unclear.

Responses did not have to give the benefits of both, but if only one, the benefits of that one needed to be well-developed.


decomposition is breaking down a problem into smaller tasks that are more manageable. Abstraction is ignoring or hiding unnecessary information so you can focus on the more important things. decomposition could help the students as they could separate the tasks between them so they can complete the game faster. For example one student could work on how the characters move and is controlled and another can work on how collisions with the walls are handled.

Abstraction could be used to simplify and focus on the most important tasks and isolate those which can be included later. This benefits the students as the more important parts of the game will be finished earlier. An example would be to make sure the character can move and that the game is playable



before adding secondary things like a main menu or the score

decomposition helps with how the tasks to complete are handleded so it can help to finish the game more quickly ~~but~~, while abstraction can help to differen Tiate which problem is more important and should be tackled next.

Using these concepts helps to polish and better tackle the development of the game as they can know what to do first how to divide their time and what ~~requires~~ requires their immediate attention.

This response scored 4 marks. The response defines decomposition and abstraction and mentions some benefits to the group of students. However, there is some repetition of points made and no reference to the program code.

If a response only defined the terms, it can be no more than Level 1. For Level 2, the response had to provide benefits to the students - the definitions may have been implied.

For Level 3, a response had to provide at least one example of where decomposition or abstraction can appear in the program code. It is likely that a good example of either was enough to gain full marks, as long as the [possibly implied] definitions and benefits were also there.