

**2015 specification**  
for the 2025 exam



# PAPER 1 EXAM RESOURCE PACK 2025

for A Level AQA Computer Science

**PYTHON<sup>3</sup> EDITION**

## - DIGITAL RESOURCE -

This pack includes paper versions of the electronic files.

Go to [zzed.uk/ProductSupport](https://zzed.uk/ProductSupport) to download the electronic files.



**POD**  
**12391**

[zigzageducation.co.uk](https://zigzageducation.co.uk)

Publish your own work... Write to a brief...  
Register at [publishmenow.co.uk](https://publishmenow.co.uk)

Follow us on Bluesky or X [@ZigZagComputing](https://twitter.com/ZigZagComputing)

# Contents

<b>Product Support from ZigZag Education .....</b>	<b>ii</b>
<b>Terms and Conditions of Use .....</b>	<b>iii</b>
<b>Teacher's Introduction .....</b>	<b>iv</b>

## **Printouts of electronic resources (for reference)**

- Code Breakdown (9 pages)
- Training Game Expressions (1 page)
- UML Class Diagram: Complete (1 page)\*\*
- UML Class Diagram: Activity (1 page)\*
- Theory Questions: Non-write-on Version (3 pages)
- Theory Questions: Write-on Version (6 pages)
- Coding Tasks (21 pages)
- Additional Tasks (Extension) (2 pages)
- Theory Questions: Mark Scheme (3 pages)\*\*
- Coding Tasks: Mark Scheme (47 pages)\*\*
- Electronic Answer Document (EAD) (3 pages)

*\* Note there are also electronic copies of the UML Diagrams ('Complete' & 'Activity' versions) provided.*

*\*\* The electronic PDF versions of these files are password-protected, so that students can only access them with your permission. Passwords can be found in the Teacher's Introduction on page iv.*

## Teacher's Introduction

**Target Clear** is a single-player game which is a cross between the 1980s game *Space Invaders* and the TV game show *Countdown*.

The user is given a list of five numbers which they can use to create a mathematical expression. The game has a list of 20 target numbers. On each turn, the user enters a mathematical expression which they are aiming to evaluate to one of the targets in the Targets list. This removes the target from the Targets list. The first five elements in the Targets list are blank – giving the user some empty space. However, after each turn the list moves one index to the left, slowly moving the targets into that empty space. If a target gets all the way to the left-hand side of the list, the game is over.

The expression entered by the user can only use the mathematical operators +, -, /, \*. The expression cannot include brackets but will correctly interpret the precedence of the accepted operators.

If the user enters an expression which evaluates to one (or more than one) target in the Targets list, that target is removed, and points are awarded to the user. The list then moves to the left.

If the user enters an expression which does not evaluate to one of the targets in the Targets list, points are deducted from the user and the list moves to the left.

This resource aims to help you get to grips with and prepare for the A Level Paper 1 examination for summer 2025, which is partly based on the **Target Clear** pre-release material.

### DIGITAL RESOURCE

Once you have downloaded the files for this resource via ([zzed.uk/ProductSupport](https://zzed.uk/ProductSupport)) you will have access to the following:



TargetClear	this folder contains all of the content (PDF/DOCX) accessible via a HTML interface
Passwords.txt	for teacher use – this file contains all of the passwords for the protected PDFs (also listed below)

\* PRINTED COPIES OF ALL THE MATERIALS IN THIS DIGITAL RESOURCE PACK ARE INCLUDED FOR REFERENCE.

**Installation:** Extract the files from the downloaded ZIP file and move the entire TargetClear folder onto a network location that is accessible for students, and provide them with a shortcut to the index.html file. All content can be accessed from this page.

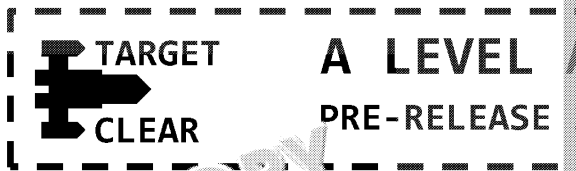
**Passwords:** All of the PDFs accessible via the *Solutions* web page are password-protected, so that students can only access them with your permission. Each password is a four-digit code, as follows:

- py02a-UML-Diagam-Complete.pdf
- py06-TheoryQuestions-MS.pdf
- py07-CodingTasks-MS.pdf

The resource pack consists of the following sections:

- **Code breakdown:** a detailed technical overview of the skeleton program, describing in detail each class and method in turn – including their purpose/function, parameters and return values. Note that this is intended as a helpful reference document only, and not as a substitute for exploring the code in a practical manner.
- **Training game expressions:** a list of expressions which evaluate to all the values in the **Targets** list using the values in the **NumbersAllowed** list. Some of these expressions use operators which are not valid in the base version of the pre-release code but will give students an opportunity to develop extension solutions and test them.
- **UML class diagram activity:** requires you to study the program and fill in the gaps with the missing class/method names, data types, associations and access levels.
- **Video:** a quick overview of the **Target Clear** game mechanics – intended as a visual aid to accompany the notes in the official AQA pre-release material.
- **Theory questions:** designed to test your understanding of the skeleton program. These questions require access to the program, but no modifications need to be made to the program. Write-on (with answer lines) and non-write-on versions are available.
- **Coding tasks:** there are 19 modification tasks to test your programming skills – as well as an additional 13 modification ideas that you may also want to try as extension tasks.
- **Solutions / Mark Schemes** for: UML Diagram Activity, Theory Questions, and Coding Tasks.

This resource is intended to supplement your teaching only. **Please read full disclaimer (p. iii) before using it.**



## Skeleton Code Breakdown

### Static Methods

Identifier / Data		Description
<b>CheckIfUserInputValid</b>		
Parameters	UserInput : String Targets : Integer List InputInRPN : String List Score : Int	This method checks if the evaluation of the expression in the Targets list and awards points accordingly. The method firstly calls the EvaluateRPN method which evaluates the user inputted expression, returning a UserInputEvaluation. The method then sets the UserInputEvaluation to True if it has a default of False. The method tests if the UserInputEvaluation could not be evaluated. If the UserInputEvaluation is found the Score is incremented by 2 and if not found it is decremented to -1 and the UserInputEvaluation is set to False. Once the loop is complete, the current score is returned.
Return values	UserInputEvaluationIsATarget : Bool Score : Int	
<b>CheckIfUserInputValid</b>		
Parameters	UserInput : String	This method uses a Regular Expression to check if the UserInput is a valid infix expression. The Regular Expression used is: $^{[0-9]+}$ To match, the UserInput parameter must be a valid mathematical expression (which can only be treated as literal characters). This entire expression must be repeated one or many times. The string must end with a valid mathematical operator. If the UserInput parameter matches the Regular Expression then it returns True, otherwise it returns False.
Return values	Bool	

INSPECTION COPY

COPYRIGHT  
PROTECTED



CheckNumbersUsedAreAllInNumbersAllowed		
Parameters	NumbersAllowed : Integer List UserInputInRPN : String List MaxNumber : Int	This method is used to test if the numbers  The method firstly creates a temporary int the NumbersAllowed list assigning copie are, by default, passed as references not list items to find them to prevent multiple list. This method removed values directly application elsewhere.  The method then iterates through the Us CheckValidNumber to confirm the elem ensure that only operands are compared subsequently checks if the operand is cor from the Temp list. If the operand is NOT because it has found an operand which c  The CheckValidNumber check does not UserInputInRPN does not meet with the than MaxNumber, the method doesn't ac
Return values	Bool	
CheckValidNumber		
Parameters	Item : String MaxNumber : Int	This method checks if a value passed to the game.
Return values	Bool	This method uses a Regular Expression integer number.  The Regular Expression used is: <code>^[0-9]+</code>  To match the Item parameter must be of Regular Expression pattern, the method the method then tests to see if ItemAsInt MaxNumber parameter. If it is, the metho method returns False.



INSPECTION COPY

COPYRIGHT  
PROTECTED





CreateTargets		
Parameters	SizeOfTargets : Int MaxTarget : Int	This method populates the <b>Targets</b> list and The method initialises the <b>Targets</b> integer indices with the value -1.
Return values	Targets : Integer List	It the <b>Score</b> is second count-controlled loop min is 4 to continue populating the list with standard pre-release game this will result
DisplayNumbersAllowed		
Parameters	NumbersAllowed : Integer List	This method is used to display all the values The method iterates through the <b>NumbersAllowed</b>
Return values		
DisplayScore		
Parameters	Score : Int	This method displays the current game <b>Score</b>
Return values	n/a	
DisplayState		
Parameters	Targets : Integer List NumbersAllowed : Integer List Score : Int	This method displays the current state of <ul style="list-style-type: none"> <li>• <b>DisplayTargets</b> – to display the contents</li> <li>• <b>DisplayNumbersAllowed</b> – to display</li> <li>• <b>DisplayScore</b> – to display</li> <li>• play the current game <b>Score</b></li> </ul>
Return values	n/a	
DisplayTargets		
Parameters	Targets : Integer List	This method is used to display all the values pipe symbol
Return values	n/a	The method iterates through the <b>Targets</b> blank space onto the screen, otherwise it



INSPECTION COPY

INSPECTION COPY



COPYRIGHT  
PROTECTED









GetNumber		
Parameters	MaxNumber : Int	This method returns a random number between 1 and MaxNumber.
Return values	Int	
<b>Main</b>		
Parameters	default	This is the main entrance point for the application. It will use a standard game with a randomly generated content list or a game with fixed content lists.
Return values	n/a	
		<p>It initialises the following variables with default values:</p> <ul style="list-style-type: none"> <li>• <b>NumbersAllowed</b> as an integer list.</li> <li>• <b>Targets</b> as an integer list.</li> <li>• <b>MaxNumberOfTargets</b> as an integer.</li> <li>• <b>MaxTarget</b> as an integer.</li> <li>• <b>MaxNumber</b> as an integer.</li> <li>• <b>TrainingGame</b> as a Boolean.</li> </ul> <p>The method asks the user if they would like to play a training game.</p> <p>If the user selects a training game, these are the default values for the game:</p> <ul style="list-style-type: none"> <li>• <b>MaxTarget</b> = 1000</li> <li>• <b>MaxNumber</b> = 1000</li> <li>• <b>TrainingGame</b> = True</li> <li>• The <b>Targets</b> list is populated with 20 random integers.</li> </ul> <p>If the user does not select a training game, these are the default values for the game:</p> <ul style="list-style-type: none"> <li>• <b>MaxTarget</b> = 10</li> <li>• <b>MaxNumber</b> = 50</li> <li>• <b>TrainingGame</b> = False</li> <li>• The <b>Targets</b> list is populated with 20 random integers inclusive.</li> </ul> <p>The method calls the <b>FillNumbers</b> method to populate the <b>NumbersAllowed</b> list and the main <b>PlayGame</b> method to start the game.</p>
		

INSPECTION COPY

COPYRIGHT  
PROTECTED



PlayGame		
Parameters	Targets : Integer List NumbersAllowed : Integer List TrainingGame : Bool MaxTarget : Int MaxNumber : Int	Initialises the following local variables with: <ul style="list-style-type: none"> <li>• Score to 0</li> <li>• GameOver to False</li> <li>• UserInput as string</li> <li>• UserInputInRPN as a list of strings</li> </ul>
Return values	n/a	These variables are then used and populated with values.
<p>The method then enters into the main application loop. The <code>GameOver</code> variable is set to <code>True</code> if the user enters a quit command. The loop operates until the <code>GameOver</code> variable is set to <code>True</code>.</p> <ul style="list-style-type: none"> <li>• Call the <code>DisplayState</code> method passing in the <code>Score</code> variable to display the current values in these variables.</li> <li>• Prompt the user to enter an infix mathematical expression into the <code>UserInput</code> variable.</li> <li>• Call the <code>CheckIfUserInputValid</code> method to check if the input is a valid infix mathematical expression.</li> <li>• If the input is valid, the <code>ConvertToRPN</code> method is called to convert the infix <code>UserInput</code> into reverse Polish notation (<code>UserInputInRPN</code>).</li> <li>• Call the <code>CheckNumbersUsedAreAllValid</code> method to check if the numbers in the <code>UserInputInRPN</code> list, <code>UserInputInRPN</code> list and the <code>NumbersUsed</code> list are all valid.</li> <li>• If all the values in the <code>UserInputInRPN</code> list are valid, the <code>CheckIfUserInputEvaluationIsATarget</code> method is called, passing in the <code>UserInputInRPN</code> list and the <code>Score</code> variable.</li> <li>• If <code>UserInputInRPN</code> evaluates to one, the <code>Score</code> variable is appropriately incremented. The <code>RemoveUserInput</code> method is called, passing in the <code>UserInput</code> variable, <code>MaxNumber</code> variable and the <code>NumbersUsed</code> list to remove the <code>UserInput</code> from the <code>NumbersUsed</code> list. The <code>Score</code> variable is then decremented if the user has successfully identified a target.</li> <li>• The method then tests to see if the <code>Score</code> variable is equal to the <code>MaxTarget</code>. If <code>GameOver</code> variable is set to <code>True</code> with the <code>Score</code> variable equal to the <code>MaxTarget</code>, the <code>Update</code> method is called with the <code>TrainingGame</code> and <code>MaxTarget</code> variables to update the <code>TrainingGame</code> and <code>MaxTarget</code> index to the left.</li> </ul> <p>If the <code>GameOver</code> variable has been set to <code>True</code> and the final <code>Score</code> are displayed on the screen.</p>		



INSPECTION COPY

COPYRIGHT  
PROTECTED



RemoveNumbersUsed		
Parameters	UserInput : String MaxNumber : Int NumbersAllowed : Integer List	This method removes any numbers from evaluation match with a target.
Return values	NumbersAllowed : Integer List	The method firstly calls the ConvertToRPN version of the expression. Although when the <code>PlayGame</code> method the <code>UserInputInRPN</code> list, by default, passed as references not by value. The method firstly calls the <code>CheckIfUserInputEvaluationIsATarget</code> method to check if the <code>UserInputInRPN</code> list, consequently <code>RemoveNumbersUsed</code> method removes the expression from the user to rebuild a new expression.  The method then iterates through the <code>UserInputInRPN</code> list and calls the <code>IsValidNumber</code> method to confirm the element is a valid number. The method also ensures that only operands are compared and checks if the operand is contained in the <code>NumbersAllowed</code> list.  Finally the method returns the <code>NumbersAllowed</code> list.
UpdateTargets		
Parameters	Targets : Integer List TrainingGame : Bool MaxTarget : Int	This method uses a count-controlled loop to backfill the list with a new value. This repeats until the list is full.
Return values	Targets : Integer List	The method firstly iterates through the <code>Targets</code> list and has the effect of moving each value one position to the right.  The method then removes the last element from the <code>Targets</code> list.  The method then uses selection on the <code>Targets</code> list to find the maximum value. If the training game and therefore the value at the end of the list. If False, the user has selected a new target. The method then adds the value in the parameter <code>MaxTarget</code> . The method then adds the value in the parameter <code>MaxTarget</code> (inclusive) and adds it to the <code>Targets</code> list.  Finally the method returns the <code>Targets</code> list.



INSPECTION COPY



INSPECTION COPY

INSPECTION COPY

COPYRIGHT  
PROTECTED



TARGET	A LEVEL A
CLEAR	PRE-RELEASE 2

## Training Game Expression

Below are expressions which will evaluate to each of the targets in the Target Number Approved list.

Most are not usable given the limitations of the pre-release base code, but they are for developing their own solutions to test:

$$68 = 2^3 + 3 + 2 + 2$$

$$23 = (8 + 2) * 2 + 3$$

$$34 = 512 / 8 / 2 + 2$$

$$119 = 512 / 8 * 2 - 3^2$$

$$9 = 3 - 2 + 8$$

$$140 = (512 / 2 + 8 * 3) / 2$$

$$82 = ((512 - 8) / 3) / 2 - 2$$

$$121 = ((512 / 8) - 2) * 2 - 3$$

$$75 = 512 / 8 + 3^2 + 2$$

$$45 = (8 - 3) * \log_2 512$$

$$43 = (\text{Concatenate } 2 \text{ and } \log_8 512) * 2$$

INSPECTION COPY

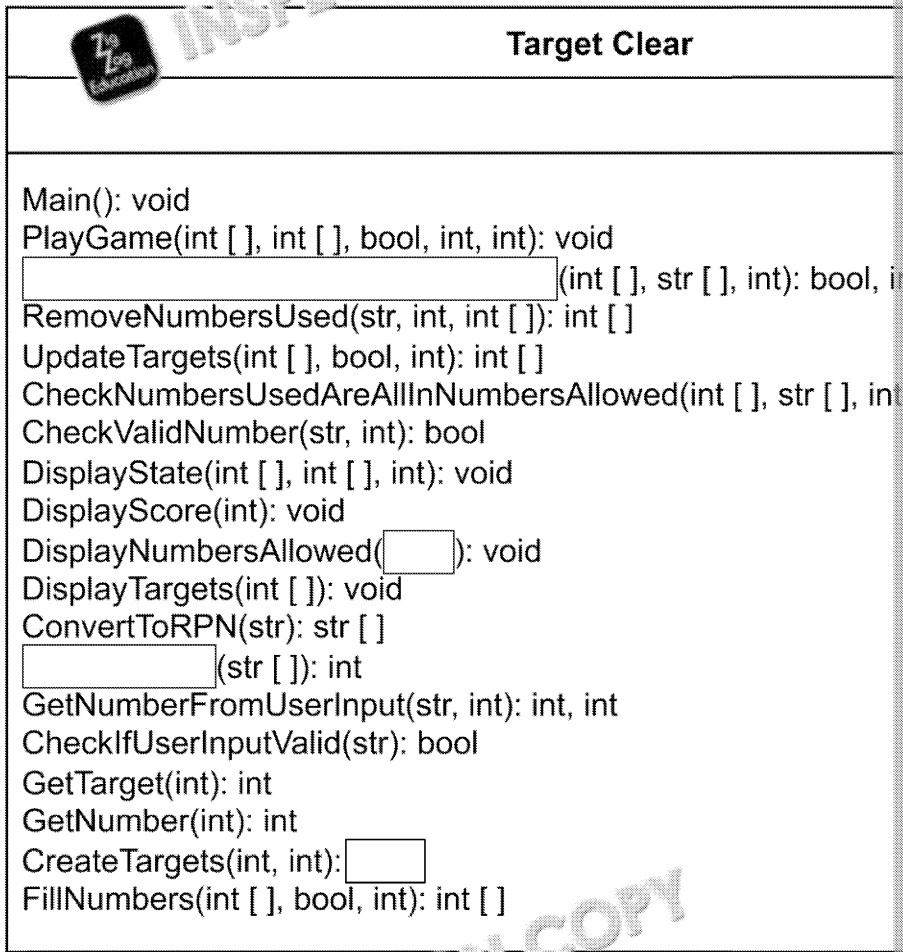
INSPECTION COPY

COPYRIGHT  
PROTECTED



# UML Class Diagram

Activity



INSPECTION COPY

COPYRIGHT  
 PROTECTED



## Theory Questions

*These questions are designed to test your understanding of the skeleton code and to the kinds of question you can expect to see in Section C of the Paper 1 examination. The sub-questions that are more than 2 marks are rarely seen in this section – the 1 mark questions are here to challenge your understanding of the code.*



These questions refer to the **Preliminary Material** and the **Skeleton Code** but **do not** require any additional programming.

**TOTAL MARKS: 57**

1. This question is about the **Main()** subroutine.
  - (a) Explain why the **Choice** variable is converted to lower case in the code.
  - (b) Explain the purpose of the **TrainingGame** variable in the program.
  
2. This question is about the **PlayGame()** subroutine. It repeatedly calls **PlayGame()**. Explain the purpose of this repeated call and how it contributes to the code.
  
3. This question is about the **RemoveNumbersUsed()** function.
  - (a) Identify what **UserInputInRPN** represents within this function.
  - (b) Explain the logic used to remove numbers from the **NumbersAllowed** list.
  
4. This question is about the function **CheckIfUserInputEvaluationIsAtLeastTarget** to modify the player's score.
  - (a) What condition needs to be met to increase the player's score?
  - (b) Why is the target set to -1 after it has been evaluated successfully?
  
5. This question is about the function **CheckValidNumber()**. The function uses a regular expression to validate user input.
  - (a) Explain the purpose of using the regular expression in this function and how the regular expression works to validate user input.
  - (b) What could happen if the regular expression pattern was changed from `[0-9+]` to `[0-9]`?
  
6. This question is about the **EvaluateRPN()** function. It evaluates expressions in Reverse Polish Notation (RPN).
  - (a) Briefly describe how Reverse Polish Notation works and how it is evaluated.
  - (b) What would happen if an invalid operation (e.g. division by zero) is attempted?

**COPYRIGHT  
PROTECTED**



## Theory Questions

*These questions are designed to test your understanding of the skeleton code and to the kinds of question you can expect to see in Section C of the Paper 1 exam. Questions that are more than 2 marks are rarely seen in this section – these more involved questions challenge your understanding of the code.*



These questions refer to the **Preliminary Material** and the **Skeleton Code** but **do not** require any additional programming.

**TOTAL MARKS: 57**

1. This question is about the **Main()** subroutine.

(a) Explain why the **Choice** variable is converted to lower case in the code.

.....

.....

(b) Explain the purpose of the **TrainingGame** variable in the program.

.....

.....

2. This question is about the **PlayGame()** subroutine. It repeatedly calls **PlayGame()**. Explain the purpose of this repeated call and how it contributes to the game.

.....

.....

.....

3. This question is about the **RemoveNumbersUsed()** function.

(a) Identify what the variable **InputInRPN** represents within this function.

.....

.....

(b) Explain the logic used to remove numbers from the **NumbersAllowed** list.

.....

.....

.....

COPYRIGHT  
PROTECTED






18. Explain how this program demonstrates the concepts of abstraction and the use of functions.

.....  
.....  
.....

19. This question is about the `UpdateTargets()` function. The function implements targets down by one position each time it is called. What is the time complexity of this function?

.....  
.....

 INSPECTION COPY

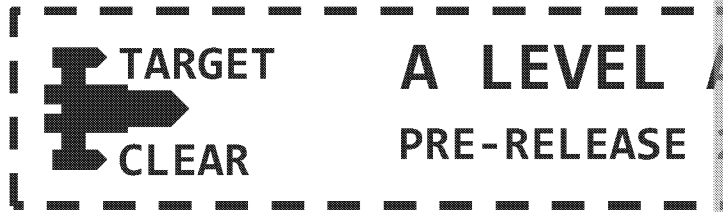
END OF QUESTIONS

INSPECTION COPY

 INSPECTION COPY

COPYRIGHT  
PROTECTED





## Programming Tasks

These questions require you to load the **Skeleton Program** and to make

*Note that any alternative or additional code changes that are deemed appropriate ensuring that it is clear where in the Skeleton Program those change*

The objective of this resource is to provide you with a selection of different questions. Some questions are more prescriptive than others in how the task should be solved, giving a range of learners. Questions which have a similar theme may use different techniques or options on how to solve problems. Some Regular Expression solutions use meta-characters beyond the AQA 7517 specification but make the solution considerably simpler. Students are encouraged to use these techniques to save coding time in the section D portion of the exam.

Students are recommended to start with a clean copy of the pre-release code for all questions in this resource. This will prevent modifications made for one question being used for a different question.

INSPECTION COPY

COPYRIGHT  
PROTECTED



## Task 1

This question extends the Skeleton Program to allow the user to end the game by entering the word "QUIT" instead of waiting until they are beaten by the **Targets**. Modify the application to allow the user to enter the word "QUIT" to end the game rather than entering an expression. The program should display the user's final score.

### What you need to do

#### Task 1.1

Update the `PlayGame` method to allow the user to enter the word "QUIT" instead of an expression. Ensure that the code does not decrement the score on that turn.

Test the user input to either play the turn if they enter an expression or quit the game and display the current score.

#### Task 1.2

Test that the changes you have made work:

- Run the Skeleton Program.
- Enter `y` to start a training game.
- Enter the expression: `8+3-2`
- Show the program correctly identifying the target 9 and awarding the user 9 points.
- When prompted for another expression, enter the word: `QUIT`
- Show the program displaying the "Game over!" message and the final score.

#### Evidence that you need to provide:

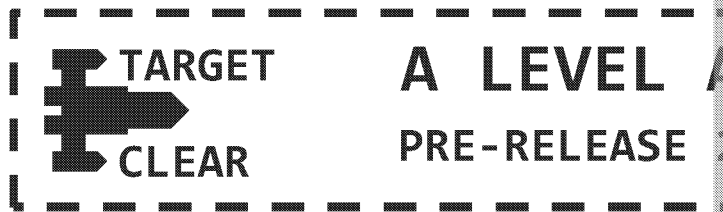
- Your PROGRAM SOURCE CODE showing the modifications to the `PlayGame` method.
- SCREEN CAPTURE(S) showing the required tests.

INSPECTION COPY

INSPECTION COPY

COPYRIGHT  
PROTECTED





## Programming Tasks (Extensions)

### Extension 1

The random game has default values of 10 for `MaxNumber` and 50 for `MaxAttempts`. Introduce a new `GameMode` enum type and add functionality for levels in the game which adjust these values. Introduce a new menu option for the user to select from the following options:

Game Mode	MaxNumber	MaxAttempts
Easy	6	30
Medium	20	100
Hard	50	100
Extreme	100	750

### Extension 2

Introduce new functionality of "Timed Challenge Mode". In this mode, the user has a limited number of attempts (e.g. 20) to identify all the targets. If the user fails to identify the targets within the allowed attempts, the game ends, and the final score is displayed. If the user achieves all targets within the allowed attempts, they are awarded an additional 50 points. Add the necessary input prompts and logic to handle this mode.

### Extension 3

Modify the application to include two `Targets` lists, enabling a two-player game. Each player should have their own `Targets` list shown on the screen at each turn, one above the other, together with the `NumbersAllowed` list. Both players should use the same `NumbersAllowed` list which should operate as a shared resource. Player 1 should identify targets in `Targets` list 1. Player 2 should identify targets in `Targets` list 2.

A player wins the game by being the first to achieve 20 points. A player loses the game if their `Targets` list reaches the first index in their `Targets` list.

### Extension 4

Modify the application to include two `NumbersAllowed` lists, enabling a competitive two-player game. Each player has their own `NumbersAllowed` list. On each turn, each player identifies a target from their own `NumbersAllowed` list which can only use values from their own list. This will evaluate to two operands. The user then enters a third expression which uses these two operands to identify a target. The user's calculation is then used to identify targets.

### Extension 5

Modify the `CheckIfUserInputEvaluationIsATarget` method to allow a difference to be awarded depending on how close the user's calculation is to a target. Award 1 point if the user's calculation is within 5 of the target and 2 points if the user's calculation is within 10 of the target.

INSPECTION COPY

**COPYRIGHT  
PROTECTED**



## **Preview of Questions Ends Here**

---

This is a limited inspection copy. Sample of questions ends here to avoid students previewing questions before they are set. See contents page for details of the rest of the resource.

Question		Suggested Solution
11	(a)	Exception handling can be useful to catch and manage runtime errors, such as invalid input errors (e.g. division by zero). It ensures that the program doesn't crash and can recover gracefully by informing the user of the issue. [1]
	(b)	Exception handling could be added in EvaluateRPN() to catch division by zero errors, allow the program to display an error message and request a new input, preventing crashing. [1]
12	(a)	The GameOver variable is set to True when the first element in the Targets list is no longer a positive integer (Targets[0] != -1). [1]
	(b)	It prevents the loop from running indefinitely, ensuring that the game ends when all relevant conditions have been met. [1]
13		Any 2 from: <ul style="list-style-type: none"> <li>The highest score would be stored in a file or a database. [1]</li> <li>At the start of each game, the file/database would be read to retrieve the previous high score. [1]</li> <li>At the end of each game, if the new score exceeds the old high score, the file/database would be updated with the new value. [1]</li> </ul>
14	(a)	CreateTargets / FillNumbers / ConvertToRPN / RemoveNumberUsed / UpdateTargets [1]
	(b)	TrainingGame [1]
	(c)	UserInput, Number [1]
	(d)	pop / append [1]
	(e)	MaxTarget / MaxNumber / MaxNumberOfTargets [1]
15		Any 2 from: <ul style="list-style-type: none"> <li>+ - means 1 or more of preceding character/sequence [1]</li> <li>[0-9]+ means 1 or more digits from 0 to 9 [1]</li> <li>([0-9]+[\\+\\-\\*\\/]) means 1 or more sequences of a number (operand) followed by an operator [1]</li> </ul>
16		Because regular expressions do not support recursion. [1] A regular expression cannot track the opening and closing of brackets / a regular expression cannot maintain a "state". [1]
17		The precedence of the current operator is compared to the precedence of the operator on top of the Operators stack. [1] While it is greater, the top of the stack is repeatedly popped onto UserInputInRPN output. [1] A final single check is carried out to compare whether the top of the stack has the same precedence as the current operator. If it has, the stack is popped once more onto the UserInputInRPN output. [1]
18		Decomposition: The problem is broken into smaller tasks, each handled by specific functions. [1] Abstraction: Don't worry about the complexity of certain tasks behind clear, high-level functions. [1]
19		On elements in the target list, n operations will be carried out. [1]

INSPECTION COPY

COPYRIGHT  
PROTECTED



## Task 19

### Coding

- Prompt to ask the user if they would like helper suggestions. [1 mark]
- Selection to branch program appropriately depending on their choice to accept helper suggestions.
- Suitable data structure to store text expressions and associated evaluations. [1 mark]
- Count-controlled loop to iterate through data structure storing text expressions and associated evaluations.
- Iterating through the NumbersAllowed list to test permutations. [1 mark]
- Rotating the NumbersAllowed list (or similar function) to test different permutations of numbers.
- Appropriately displaying the combination of text expressions and associated evaluations on the screen.
- Use of recursion to try combinations. [1 mark]
- Only storing combinations for targets which have not already been identified. [1 mark]
- Correctly generating expressions which use division to ensure they evaluate to an integer. [1 mark]
- Testing expressions to ensure they correctly follow BIDMAS if needed (required for expressions built up step by step).
- Generate expressions which can use the four mathematical operators: + - / \* [1 mark]
- Storage of expression with associated evaluation. [1 mark]

### Teacher Notes:

*This functionality could be completed using iteration. Marks should be awarded for techniques, but full marks require recursion.*

*Because the expression is built up step by step, it must be tested at each stage because the impact of BIDMAS is not obvious until the end.*

### Example Solution

Modification of the PlayGame method:

```
while not GameOver:
    DisplayState(Targets, NumbersAllowed, Score)
    #CHANGE
    UserChoice = input("Would you like helper suggestions: Y/N ").upper()
    if UserChoice == "Y":
        Temp = []
        PossibleSolutions = {}
        for Item in NumbersAllowed:
            Temp.append(Item)
            for i in range(5):
                TestSolutions = GenerateEvaluations(Temp, Targets)
                for key, value in TestSolutions.items():
                    if key not in PossibleSolutions:
                        PossibleSolutions[key] = value
            Temp.append(Temp[0])
```

INSPECTION COPY

COPYRIGHT  
PROTECTED



```

        del Temp[0]
    print()
    for key, value in PossibleSolutions.items():
        print(f"{key} can be calculated using the expression: {value}")
    print()
#END CHANGE
UserInput = input("Enter an expression: ")
print()

```

Creation of new GenerateEvaluation method and associated helper method):

```

#CHANGE
def GenerateEvaluations(NumbersAllowed, Targets):
    PossibleExpressions = {}
    GenerateEvaluationsHelper(NumbersAllowed, Targets, 0, NumbersAllowed[0], PossibleExpressions)
    return PossibleExpressions

def GenerateEvaluationsHelper(NumbersAllowed, Targets, Index, CurrentResult, PossibleExpressions):
    if Index == len(NumbersAllowed)-1:
        #Because the recursion calculates expressions step by step rather than all at once
        #the new code needs to test the end result using RPN evaluator to ensure it is valid
        if EvaluateRPN(ConvertToRPN(CurrentExpression)) in Targets and EvaluateRPN(ConvertToRPN(CurrentExpression)) not in PossibleExpressions:
            PossibleExpressions[EvaluateRPN(ConvertToRPN(CurrentExpression))] = CurrentExpression
        return
    NextNumber = NumbersAllowed[Index + 1]
    GenerateEvaluationsHelper(NumbersAllowed, Targets, Index + 1, CurrentResult + NextNumber,
    Expression}*{NextNumber}")
    if NextNumber != 0:
        if float(CurrentResult / NextNumber) - math.floor(float(CurrentResult / NextNumber)) == 0:
            GenerateEvaluationsHelper(NumbersAllowed, Targets, Index + 1, math.floor(float(CurrentResult / NextNumber)) * NextNumber,
    Expressions, f"{CurrentExpression}/{NextNumber}")
        GenerateEvaluationsHelper(NumbersAllowed, Targets, Index + 1, CurrentResult - NextNumber,
    Expression}-{NextNumber}")
        GenerateEvaluationsHelper(NumbersAllowed, Targets, Index + 1, CurrentResult * NextNumber,
    Expression}*{NextNumber}")
        GenerateEvaluationsHelper(NumbersAllowed, Targets, Index + 1, CurrentResult / NextNumber,
    Expression}/{NextNumber}")
#END CHANGE

```

INSPECTION COPY

**COPYRIGHT  
PROTECTED**





## Testing

- Show the program displaying the suggested valid expressions for targets. [1 mark]

```
Enter y to play the training game, anything else to play a random
| | | | | |8|8|17|12|13|34|11|32|35| |8| |6|40|32|
Numbers available: 1 7 10 6
Current target: 38
Would you like helper suggestions: Y/N
y
38 can be calculated using the expression: 1*7*6-10+6
17 can be calculated using the expression: 1*7+6+10-6
6 can be calculated using the expression: 1+7-6+10-6
Enter an expression: |
```



INSPECTION COPY

INSPECTION COPY

COPYRIGHT  
PROTECTED



## **Preview of Answers Ends Here**

---

This is a limited inspection copy. Sample of answers ends here to stop students looking up answers to their assessments. See contents page for details of the rest of the resource.

Name

ZigZag Education supporting

# A Level AQA Computer Science Paper

## Summer 2025



### Electronic Answer Document (EAD)

#### Instructions

- Enter your name in the box at the top of this page
- Answer **all** questions by entering your answers into this document
- Remember to **save** this document regularly
- Save and print this document and any additional pages
  
- Answer **all** questions
- The marks available for each question are shown in brackets
  
- You will need:
  - access to a computer
  - access to a printer
  - access to appropriate software
  - electronic copies of the required skeleton code
  - EAD (Electronic Answer Document)

Total marks:

INSPECTION COPY

COPYRIGHT  
PROTECTED



# Exam-style Questions

Answer all questions. Remember to save this document

Q	Answer
1	(a)
	(b)
2	
3	(a)
	(b)
4	(a)
	(b)
5	(a)
	(b)
6	(a)
	(b)
7	
8	(a)
	(b)
9	(a)
	(b)
10	(a)
	(b)
11	(a)
	(b)
12	(a)
	(b)
13	
14	(a)
	(b)
	(c)
	(d)
	(e)
15	
16	
17	
18	
19	

INSPECTION COPY

COPYRIGHT  
PROTECTED



# Exam-style Programming Tasks

Answer all questions. Remember to save this document

Q	Answer
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	

INSPECTION COPY

COPYRIGHT  
PROTECTED

