Zig Zag Education

Computer Science A Level | AQA | 7517

2015 specification for the 2025 exam

TARGET

CLEAR

PAPER 1 EXAM RESOURCE PACK 2025

for A Level AQA Computer Science

JAVA EDITION

- DIGITAL RESOURCE -

This pack includes paper versions of the electronic files.

Go to **zzed.uk/ProductSupport** to download the electronic files.

POD 12390

zigzageducation.co.uk

Contents

Printouts of electronic resources (for reference)

** 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**) 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:

🗎 `j02a-UML-Diagam-Complete.pdf`
🗎 `j06-TheoryQuestions-MS.pdf`
🗎 `j07-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.**

TARGET CLEAR A LEVEL AQ

PRE-RELEASE 20

Skeleton Code Breakdo

Static Methods

Identifier / Data		
checkIfUserInputEvaluationIsATarget		
Parameters	targets : Integer List userInputInRPN : String List score : IntWrapper	This method checks if the evaluation of in the targets list and awards points ap The method firstly calls the evaluateRP evaluates the user inputted expression, userInputEvaluation. The method then sets the userInputEv has a default of false. The method tests if the userInputEvalu userInputInRPN could not be evaluate method performs a count-controlled loo targets. The loop compares the userInp is found, the score value is incremented matched is set to 1 and the userInputE Once the loop is complete, the current s
Return values	userInputEvaluationIsATarget : Bool	
checkIfUserInputValid		
Parameters	userInput : String	This method uses a Regular Expression infix expression. The Regular Expressio The Regular Expression used is: ^([0-9] To match, the userInput parameter mu mathematical operator which can only b treated as literal characters). This entire one or many times. The string must enc If the userInput parameter matches the otherwise it returns false.
Return values	Bool	

checkNumbersUsedAreAllInNumbersAllowed		
Parameters	numbersAllowed : Integer List userInputInRPN : String List maxNumber : Int	This method is used to test if the number The method firstly creates a temporary the numbersAllowed list assigning co lists are by default passed as referenc comparison list when it finds them to pr numbersAllowed list. If the method re would impact the application elsewhere The method then iterates through the u checkValidNumber to confirm the elem to ensure that only operands are compa subsequently checks if the operand is c from the temp list. If the operand is NO it has found an operand which cannot b The checkValidNumber check does n userInputInRPN does not meet with th greater than maxNumber, the method
Return values	Bool	

checkValidNumber		
Parameters	item : String maxNumber : Int	This method checks if a value passed t This method uses a Regular Expressio integer number. The Regular Expression used is: ^[0-9] To match, the item parameter must be Regular Expression pattern, the metho itemAsInteger. The method then tests equal to the maxNumber parameter. If met, the method returns false.
Return values	Bool	

convertToRPN		
Parameters	userInput : String	This method converts the infix expressi a version of the shunting yard algorithm Initialises the following local variables: • position to 0 (as an intWrapper st • precedence to HashMap of type < with an associated value. Multiplica and Subtraction. This is used to all *not recognise Brackets or Indices,* • operand as an integer. This uses t in the infix notation. • userInputInRPN as a list of strings casted as a string. • operators as a list of strings. This userInput expression. The method then enters a condition-con operand is updated using the getNumb notation. The position object is passed this object within that method as it iterat method. The updated operand is apper the expression (assuming it is valid) mu If the position variable is less than the le operands in the expression which have n has just extracted an operand from the e operator prior to it, stores this in the varia incrementa[illegible] the operators list by pop HashMap to compare their worth. If the v than the currentOperator, it is added to The currentOperator is then added to th Division functions are added to the userI If the position variable is not less than operators from the string have been ext popping values from the back of the list The method then returns the completed
Return values	userInputInRPN: String List	

createTargets		
Parameters	sizeOfTargets : Int maxTarget : Int	This method populates the targets list The method initialises the targets integ five indices with the value -1. It then uses a second count-controlled l minus 5 to continue populating the list w standard pre-release game this will resu
Return values	targets : Integer List	

displayNumbersAllowed		
Parameters	numbersAllowed : Integer List	This method is used to display all the v The method iterates through the numb
Return values		

displayScore		
Parameters	score : Int	This method displays the current game
Return values	n/a	

displayState		
Parameters	targets : Integer List numbersAllowed : Integer List score : Int	This method displays the current state • displayTargets – to display the co • displayNumbersAllowed – to disp • displayScore – to display the curr
Return values	n/a	

displayTargets		
Parameters	targets : Integer List	This method is used to display all the v pipe symbol | The method iterates through the targe blank space onto the screen, otherwise
Return values	n/a	

evaluateRPN		
Parameters	userInputInRPN : String List	This method evaluates the RPN versior evaluates to an integer (positive or nega This method initialises a string list s. The controlled loop to iterate through the use The method iterates through the userIn (+-*/' and adding elements which are no from userInputInRPN and pushing ther number values from the start of the pos userInputInRPN list, the loop stops, an to the variables num2 and num1 (esse doubles to allow float division to be perf operator at the start of the userInputIn operation. The result of the operation is userInputInRPN is removed (essentiall the next evaluation. This process is repeated until the userI been evaluated and the list s only now The method then subtracts a truncated evaluates to 0.0, then the result must ha result cast as an integer is returned. If n has evaluated to a decimal and therefor
Return values	Int	

fillNumbers		
Parameters	numbersAllowed : Integer List trainingGame : Bool maxNumber : Int	This method repopulates the numbersA If the trainingGame parameter is true, pre-populated list with the values 2, 3, 2 values in the numbersAllowed list on e If the trainingGame parameter is false, condition-controlled loop to append valu to get a new in-range target until the list
Return values	numbersAllowed : Integer List	

getNumber		
Parameters	maxNumber : Int	This method returns a random number
Return values	Int	

getNumberFromUserInput		
Parameters	userInput : String position : IntWrapper	This method is used to extract numbers converted into postfix. The method initially instantiates an emp The method iterates through the userIn position parameter to set the index of w object, therefore changes made to its va checked using a Regular Expression to onto the number variable. This techniqu delimiter. If a character found does not r sets the moreDigits variable to false, ex the length of the userInput string, mear If the number variable is an empty strin method returns -1. If the number variab
Return values	Int	
getTarget		
Parameters	maxTarget : Int	This method returns a random number
Return values	Int	
getNumber		
Parameters	maxNumber : Int	This method returns a random number
Return values	Int	

main		
Parameters	default	This is the main entrance point for the a use a standard game with a randomly g game with fixed content lists. It initialises the following variables with • numbersAllowed as an integer lis • targets as an integer list • maxNumberOfTargets as an integ • maxTarget as an integer • maxNumber as an integer • trainingGame as a Boolean The method asks the user if they would If the user selects a training game, thes in the game: • maxTarget = 1000 • maxNumber = 1000 • trainingGame = true • The targets list is populated with 2 If the user does not select a training ga use later in the game: • maxTarget = 10 • maxNumber = 50 • trainingGame = false • The targets list is populated with 2 maxTarget) inclusive. This method calls the fillNumbers meth main playGame method to start the ga
Return values	n/a	

playGame		
Parameters	targets : Integer List numbersAllowed : Integer List trainingGame : Bool maxTarget : Int maxNumber : Int	Initialises the following local variables w • score to 0 • gameOver to false • userInput as a string • userInputInRPN as a list of strings These variables are then used and pop The method then enters into the main a gameOver variable. The loop operates • Call the displayState method pass *intWrapper static object)* variables t • Prompt the user to enter an infix m userInput variable. • Call the checkIfUserInputValid m • If the input is valid, the convertToF converts the infix userInput into R userInputInRPN. • Call the checkNumbersUsedAreA list, userInputInRPN list and the m • If all the values in the userInputInF checkIfUserInputEvaluationIsATa userInputInRPN list and the score • If userInputInRPN evaluates to on appropriately incremented. The rer userInput variable, maxNumber v used in a successful target match e method is then called, passing in th maxNumber variables to backfill th • The score value is then decremen successfully identified a target. • The method then tests to see if the gameOver variable is set to true w the targets list is not -1, the updat with the trainingGame and maxTa index to the left. If the gameOver variable has been set and the final score are displayed on th
Return values	n/a	

removeNumbersUsed		
Parameters	userInput : String maxNumber : Int numbersAllowed : Integer List	This method removes any numbers fro evaluation match with a target. The method fir[illegible] calls the convertToF version [illegible] expression. Although whe pl[illegible]Game [illegible]thod the userInputInRPN by [illegible]ult, passed as references not by checkIfUserInputEvaluationIsATarge userInputInRPN list, consequently rem expression from the user to rebuild a ne The method then iterates through the u checkValidNumber to confirm the elem to ensure that only operands are compa checks if the operand is contained in th from the numbersAllowed list.
Return values	n/a	

updateTargets		
Parameters	targets : Integer List trainingGame : Bool maxTarget : Int	This method uses a count-controlled loo backfill the list with a new value. This re The method firstly iterates through the t This has the effect of moving each valu The method then removes the last elem The method then uses selection on the training game and therefore the value a the end of th[illegible] [illegible]f false, the user has passin[illegible] pa[illegible]meter maxTarget. T m[illegible]xTa[illegible] (i[illegible]clusive) and adds it to the
Return values	n/a	

Static Class: IntWrapper

	Identifi[illegible]	
<<constructo[illegible]		
Parameters	[illegible]alValue : Int	This static class is used to wrap an inte within the application. This technique is in the convertToRPN method to create Using this technique has the similar effe by reference, as used in the other versi

TARGET CLEAR

A LEVEL A

PRE-RELEASE 2

Training Game Expression

Below are expressions which will evaluate to each of the targets in the Targ
Numbers Allowed list.

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

68 = 512/8+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 = (Concatenate 2 and $\log_8$ 512)*

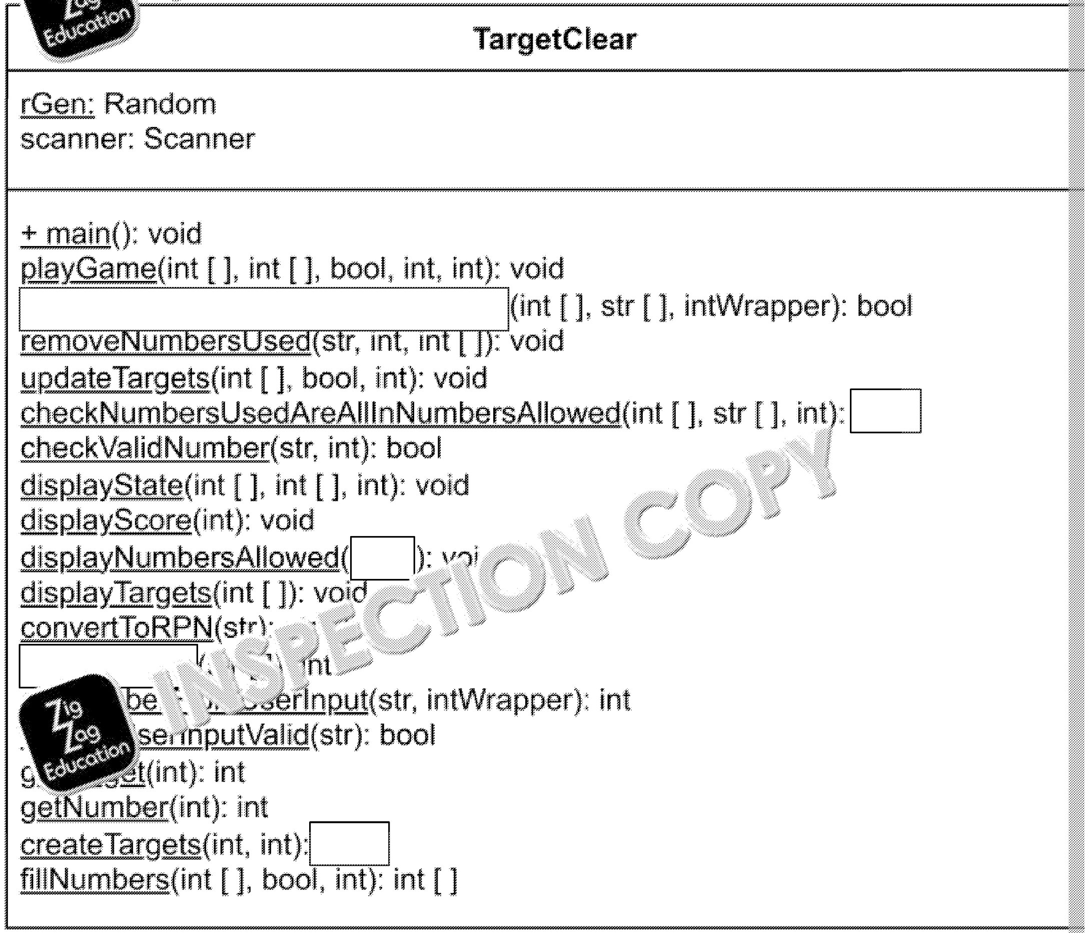
TARGET CLEAR

A LEVEL

PRE-RELEAS

UML Class Diagr

Activity



TargetClear

rGen: Random
scanner: Scanner

+ main(): void
playGame(int [ ], int [ ], bool, int, int): void
□(int [ ], str [ ], intWrapper): bool
removeNumbersUsed(str, int, int [ ]): void
updateTargets(int [ ], bool, int): void
checkNumbersUsedAreAllInNumbersAllowed(int [ ], str [ ], int): □
checkValidNumber(str, int): bool
displayState(int [ ], int [ ], int): void
displayScore(int): void
displayNumbersAllowed(□): voi
displayTargets(int [ ]): void
convertToRPN(str):
□([illegible]) int
[illegible]be[illegible]serInput(str, intWrapper): int
[illegible]serInputValid(str): bool
g[illegible]et(int): int
getNumber(int): int
createTargets(int, int): □
fillNumbers(int [ ], bool, int): int [ ]

TARGET CLEAR

A LEVEL A

PRE-RELEASE 2

Theory Questions

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

These questions refer to the **Preliminary Material** and the **Sk** but **do not** require any additional programmin

TOTAL MARKS: 57

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

   (a) Explain why the **choice** variable is converted to lower case in the

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

2. This question is about the **playGame()** subroutine. It repeatedly calls d

   Explain the purpose of this repeated call and how it contributes to the g

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 **numbersAllo**

4. This question is about the function **checkIfUserInputEvaluationIsATa** 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 u

   (a) Explain the purpose of using the regular expression in this functio expression works to validate user input.

   (b) What could happen if the regular expression pattern was changed the **+** character?

6. This question is about the **evaluateRPN()** function. It evaluates expres Notation (RPN).

   (a) Briefly describe how Reverse Polish Notation works and how it can

   (b) What would happen if an invalid operation (e.g. division by zero) is a

TARGET CLEAR

A LEVEL A

PRE-RELEASE 2

Theory Questions

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

These questions refer to the **Preliminary Material** and the **Sk** but **do not** require any additional programmin

TOTAL MARKS: 57

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

(a) Explain why the **choice** variable is converted to lower case in the

....

....

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

....

....

2. This question is about the **playGame()** subroutine. It repeatedly calls d

Explain the purpose of this repeated call and how it contributes to the g

....

....

....

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 **numbersAllo**

....

....

....

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

....

....

....

19. This question is about the **updateTargets()** function. The function imple targets down by one position each time it is called. What is the time com

....

....

END OF QUESTIONS

TARGET CLEAR

A LEVEL A

PRE-RELEASE 2

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 que questions. Some questions are more prescriptive than others in how the task sh range of learners. Questions which have a similar theme may use different techni options on how to solve problems. Some Regular Expression solutions use meta beyond the AQA 7517 specification but make the solution considerably simpler. S these techniques to save coding time in the section D portion

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

Task 1

This question extends the Skeleton Program to allow the user to end the g wait until they are beaten by the targets. Modify the application to allow th “QUIT” to end the game rather than entering an expression. The program s final score.

What you need to do

Task 1.1

Update the playGame method to allow the user to enter the word “QUIT” i 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 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 u
- When prompted for another expression, enter the word: `QUIT`
- Show the program displaying the “Game over!” message and the final

Evidence that you need to provide:

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

TARGET CLEAR

A LEVEL A

PRE-RELEASE 2

Programming Tasks (Extensi

Extension 1

The random game has default values of 10 for MaxNumber and 50 for Ma functionality for levels in the game which adjust these values. Introduce a n user to select from the following options

Game Mode	MaxNumber	
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 u attempts (e.g. 20) to identify all the targets. If the user fails to identify the ta attempts, the game ends, and the final score is displayed. If the user achie additional 50 points. Add the necessary input prompts and logic to handle t

Extension 3

Modify the application to include two Targets lines, enabling a two-player g shown on the screen at each turn, one above the other, together with the N players should use the same NumbersAllowed list which should operate a Player 1 should identify targets in Targets list 1. Player 2 should identify ta

A player wins the game by being the first to achieve 20 points. A player los targets reaches the first index in their Targets list.

Extension 4

Modify the application to include two NumbersAllowed lists, enabling a co Each player has their own NumbersAllowed list. On each turn, each playe which can only use values from their own list. This will evaluate to two ope then enter a third expression which uses these two operands to identify a t together to identify targets.

Extension 5

Modify the CheckIfUserInputEvaluationIsATarget method to allow a diffe awarded depending on how close the user's calculation is to a target. Awa target. Award 3 points if the user's calculation is within 5 of the target and 2 calculation is within 10 of the target.

Preview of Questions Ends Here

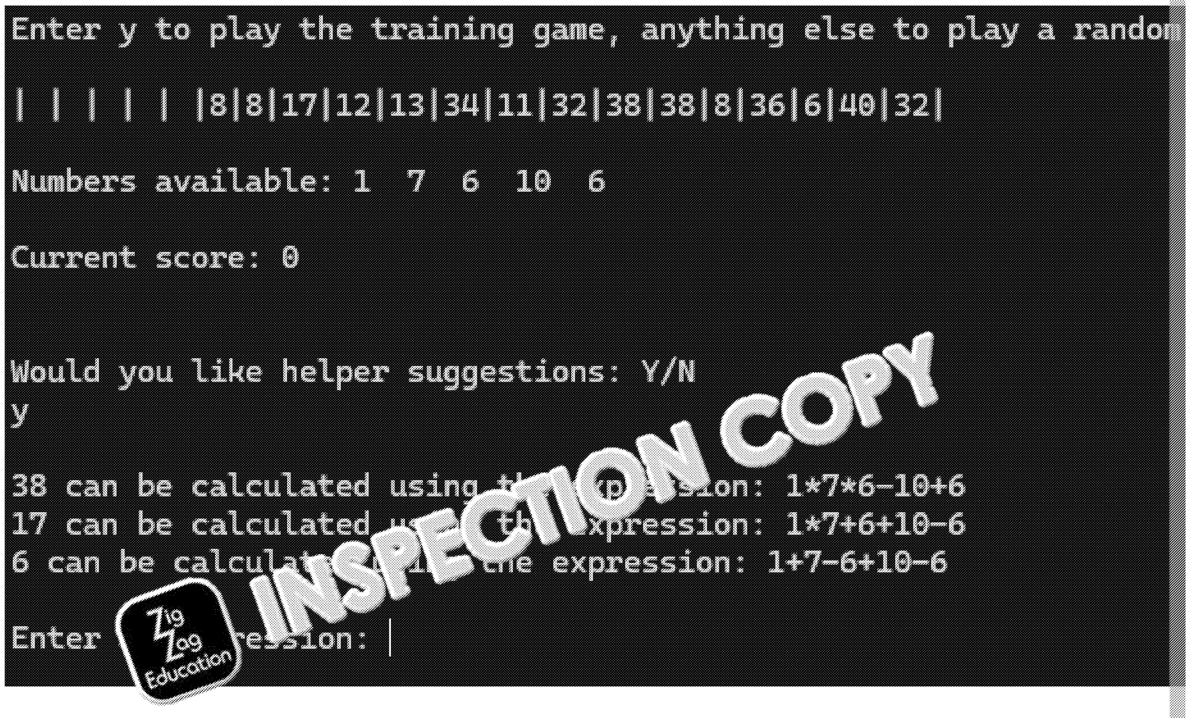
This is a limited inspection copy. Sample of questions ends here to avoid students previewing questions before they are set. See contents page for details of the rest of the resource.

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 gra informing the user of the issue. [1]
	(b)	Exception handling could be added in evaluateRPN() to catch division by zero errors, allow program to display an error message and request a new input without crashing. [1]
12	(a)	The gameOver variable is set to true when the first target in the Targets list is no longer ava 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: • The highest score could be stored in a file or a database. [1] • [illegible] game, the file/database would be read to retrieve the previous high s • [illegible] each game, if the new score exceeds the old high score, the file/database would [illegible] the new value. [1]
14	(a)	createTargets / fillNumbers / convertToRPN / removeNumberUsed / updateTargets [1
	(b)	trainingGame [1]
	(c)	userInput, number [1]
	(d)	remove / add [1]
	(e)	maxTarget / maxNumber / maxNumberOfTargets [1]
15		Any 2 from: • `+` - means 1 or more of preceding character/sequence [1] • `[0-9]+` means 1 or more digits from 0 to 9 [1] • `([0-9]+[\\+\\-\\*\\/])+` means 1 or more sequences of a number (operand) follo an operator [1]
16		Because regular expressions do not support recursion. [1] A regular expression cannot track the opening and closing of brackets / a regular expressio of "state". [1]
17		The precedence of the current operator is compared to the precedence of the operator on t operators stack. [1] While it is greater, the top of the stack is repeatedly popped onto userInputInRPN output. [A final single check is carried out to [illegible] whether the top of the stack has the same pre current operator. If it has, the stack is popped once more onto the userInputInRPN output.
18		Decomposition: The program is broken into smaller tasks, each handled by specific functior Abstraction: [illegible] by hiding the complexity of certain tasks behind clear, high-level [illegible] specific roles. [1]
19		[illegible] elements in the target list, n operations will be carried out. [1]

```
                generateEvaluationsHelper(numbersAllowed, targets, index + 1, cu
currentExpression + "/" + nextNumber);
            }
        }
        generateEvaluationsHelper(numbersAllowed, targets, index + 1, currentRes
currentExpression + "+" + nextNumber);
        generateEvaluationsHelper(numbersAllowed, targets, index + 1, currentRes
currentExpression + "-" + nextNumber);
    }
    // END CHANGE
```

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|38|38|8|36|6|40|32|

Numbers available: 1  7  6  10  6

Current score: 0


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 your expression: 
```

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 Pap

Summer 2025

TARGET CLEAR A LEVEL AQA PRE-RELEASE 20

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:

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		

Exam-style Programming Task

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	