Zig Zag Education

Computer Science A Level | AQA | 7517

2015 specification for the 2025 exam

PAPER 1 EXAM RESOURCE PACK 2025

for A Level AQA Computer Science

VB.NET EDITION

- DIGITAL RESOURCE -

This pack includes paper versions of the electronic files.

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

POD 12392

zigzageducation.co.uk

Contents

Product Support from ZigZag Education ..... **ii**

Terms and Conditions of Use ..... **iii**

Teacher's Introduction ..... **iv**

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**) 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:

- `vb02a-UML-Diagam-Complete.pdf`
- `vb06-TheoryQuestions-MS.pdf`
- `vb07-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.

TARGET CLEAR

A LEVEL

PRE-RELEASE

Sk[illegible]ch Code Breakdo

Static Methods

Identifier / D[illegible]	
CheckIfUserInp[illegible]Eval[illegible]IsA[illegible]rget	
Parameters	**[illegible]get[illegible]** : Integer List **[illegible]rInputInRPN** : String List **Score** : Int
Return values	**UserInputEvaluationIsATarget** : Bool
	This method checks if the evaluation of in the **Targets** list and awards points a[illegible] The method firstly calls the **EvaluateR[illegible]** evaluates the user inputted expression, **UserInputEvaluation**. The method then sets the **UserInputEv[illegible]** variable has a default of False. The method tests if the **UserInputEval[illegible]** **UserInputInRPN** could not be evaluate method performs a count-controlled loo targets. The loop compares the **UserIn[illegible]** match is found the **Score** is incremente matched is set to 1 and the **UserInput[illegible]** Once th[illegible]s c[illegible]mplete, the current s[illegible]
CheckIfUserInputValid	
Parameters	**UserInput** : String
Return values	Bool
	This method uses a Regular Expressio 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 end If the **UserInput** parameter matches the otherwise it returns False.

CheckNumbersUsedAreAllInNumbersAllowed		
Parameters	**NumbersAllowed** : Integer List **UserInputInRPN** : String List **MaxNumber** : Int	This method is used to test if the numbe The method firstly creates a temporary the **Numbers** **ved** list assigning co lists are **b** **au** passed as referenc co **pa** **co** **t** when it finds them to pr **Nu** **rsAllowed** list. If the method re would impact the application elsewhere The method then iterates through the **U** **CheckValidNumber** to confirm the ele to ensure that only operands are compa subsequently checks if the operand is c from the **Temp** list. If the operand is NC because it has found an operand which 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, th **'** parameter must be Regula **Ex** **ssi** pattern, the metho **Ite** **nAs** **ger**. The method then tests **equ** to the **MaxNumber** parameter. I 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
Return values	**UserInputInRPN**: String List	Initialises the fo ving local variables: • **Po** o 0. This is used to ident • **Pr** erce to Dictionary of type ith an associated value. Multiplica and Subtraction. This is used to al *not recognise Brackets or Indices.* • **Operand** as an integer. This uses number in the infix notation. • **UserInputInRPN** as a list of string casted as a string. • **Operators** as a list of strings. This **UserInput** expression. The method then enters a condition-co **Operand** is updated using the **GetNum** notation. The **Position** variable is passe variable within that method as it iterates method. The updated **Operand** is appe the expression (assuming it is valid) mu If the **Position** variable is less than the l operands in the expression which have has just extracted an operand from the e operator pric stores this in the varia incr ta th **Operators** list by po di ona compare their worth. If the tha the **CurrentOperator**, it is added to The **CurrentOperator** is then added to t Division functions are added to the **User** If the **Position** variable is not less than operators from the string have been ex popping values from the back of the list The method then returns the completed

CreateTargets		
Parameters	SizeOfTargets : Int MaxTarget : Int	This method populates the Targets lis The method initialises the Targets inte five indices wit e value -1. It t se: second count-controlled m us 5 continue populating the list tandard pre-release game this will res
Return values	Targets : Integer List	
DisplayNumbersAllowed		
Parameters	Numbers : eger List	This method is used to display all the v The method iterates through the Numb
Return value		
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 c • DisplayNumbersAllowed – to dis • DisplayScore – to display the cur
Return values	n/a	
DisplayTargets		
Parameters	Targets : Integer List	Th method is used to display all the v ipe symbol |
Return values	n/a	
		The method iterates through the Targe blank space onto the screen, otherwise

EvaluateRPN		
Parameters	**UserInputInRPN** : String List	This method evaluates the RPN versio evaluates to an integer (positive or neg This method ini es a string list **S**. The controll o iterate through the **Use** Th m iterates through the **UserI** '+-*/' and adding elements which are n from **UserInputInRPN** and pushing the number values from the start of the pos **UserInputInRPN** list, the loop stops, a 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 (essentia for the next evaluation. This process is repeated until the **User** been evaluated and the list **S** only now The method then subtracts a truncated v evaluates to 0.0, then the result must ha result cast as an integer is returned. If n evaluated to a decimal and therefore ca
Return values	Int	
FillNumbers		
Parameters	**NumbersAllowed** : Integer List **TrainingGame** : Bool **MaxNumber** : Int	This method re ulates the **Numbers** If the **T Ga e** parameter is True a e-p ated list with the values 2, 3 val in the **NumbersAllowed** list on
Return values	**NumbersAllowed** : Integer List	
		If the **TrainingGame** parameter is Fals condition-controlled loop to append val to get a new in-range target until the lis
GetNumber		
Parameters	**Number** : Int	This method returns a random number
Return values	Int	

GetNumberFromUserInput		
Parameters	**UserInput** : String **Position** : Int	This method is used to extract numbers converted into postfix. The method ini[illegible]y instantiates an emp Th[illegible] no[illegible]erates through the **UserI** **Po[illegible]iti[illegible]** parameter to set the index of reference rather than by value, therefo finishes. Each character is checked us 9. If it is, it is concatenated onto the **Nu** multiple digit numbers without a delimit Expression, it must be an operator whic loop also exits if the **Position** variable iterated to the end of the string. If the **Number** variable is an empty stri method returns -1. If the **Number** variat
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
Return values	n/a

This is the main entrance point for the a
use a standard game with a randomly g
training game with fixed content lists.

It initiali[illegible] fo[illegible]wing variables with
- **Nu[illegible]rsAllowed** as an integer lis
- **[illegible]rgets** as an integer list.
- **MaxNumberOfTargets** as an inte
- **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

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
  MaxTa[illegible] clusive.

Th[illegible] me[illegible]hc[illegible] calls the **FillNumbers** met
ma[illegible]**layGame** method to start the ga

PlayGame		
Parameters	**Targets** : Integer List **NumbersAllowed** : Integer List **TrainingGame** : Bool **MaxTarget** : Int **MaxNumber** : Int	Initialises the following local variables w • **Score** to 0 • **GameOver** t False. • **Use** a string. • **Use** **InRPN** as a list of string The variables are then used and pop The method then enters into the main a **GameOver** variable. The loop operates • Call the **DisplayState** method pas display the current values in these • Prompt the user to enter an infix m variable. • Call the **CheckIfUserInputValid** m • If the input is valid, the **ConvertTo** converts the infix **UserInput** into re **UserInputInRPN**. • Call the **CheckNumbersUsedAre** **NumbersAllowed** list, **UserInput** • If all the values in the **UserInputIn** **CheckIfUserInputEvaluationIsA** **UserInputInRPN** list and the **Sco** rather than as a value. • If **UserInputInRPN** evaluates to o appropriately incremented. The **Re** **UserIn** iable, **MaxNumber** v successful target match m is then called, passing in t **MaxNumber** variables to backfill t • The **Score** variable is then decrem successfully identified a target. • The method then tests to see if the **GameOver** variable is set to True the **Targets** list is not -1, the **Upda** together with the **TrainingGame** a **Targets** list one index to the left. If the **GameOver** variable has been set over!" and the final **Score** are displayed
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 fi calls the **ConvertTo** version exp ession. Although wh **Pl gGa** method the **UserInputInRP** by ault, passed as references not by **CheckIfUserInputEvaluationIsATarge** **UserInputInRPN** list, consequently **Re** expression from the user to rebuild a ne The method then iterates through the **U** **CheckValidNumber** to confirm the ele 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 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 st. If False, the user has pa in parameter **MaxTarget**. T M T g t (inclusive) and adds it to the
Return values	n/a	

TARGET CLEAR

A LEVEL A

PRE-RELEASE 2

Training Game Expression

Below are expressions which will evaluate to each of the targets in the Targe
Numbe[illegible]ed list.

Most are not usable given the limitations of the pre-release base code, but the
fo[illegible]eveloping their own solutions to test:

```
68 = [illegible]/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)*log2512
43 = (Concatenate 2 and log8 512)*
```

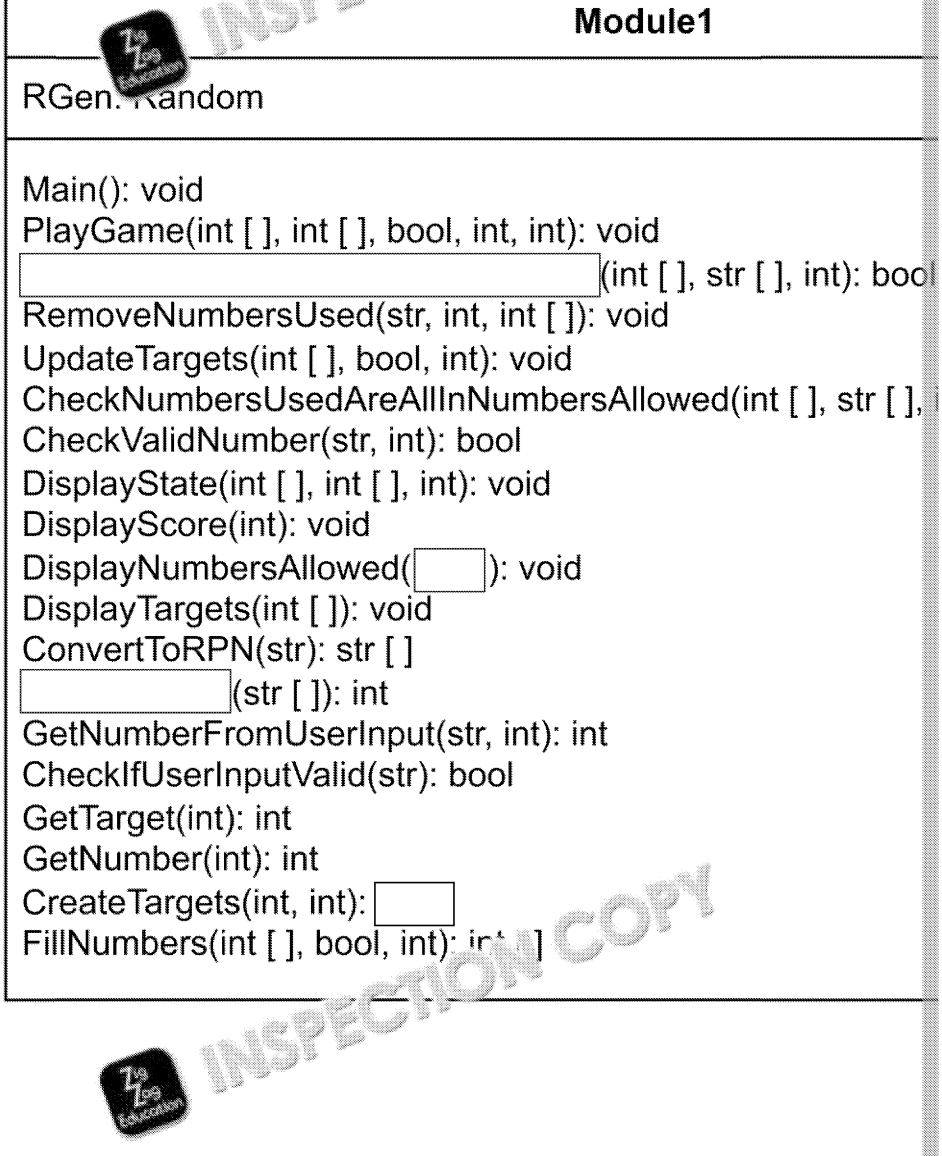
TARGET CLEAR

A LEVEL A

PRE-RELEASE

UML Class Diagra

Activity



Zig Zag Education

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 invo 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

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

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 **CheckIfUserInputEvaluationIsAT** 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

(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 invo 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

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

.....

.....

.....

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

(a) Identify what **InputInRPN** 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 co[illegible]nges that are deemed appropriate [illegible] ensuring that it is clear where in the Skeleton Program those change

The obj[illegible] 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 ga
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 Ma[illegible]er and 50 for Ma[illegible] functionality for levels in the game which a[illegible]ust t[illegible]e values. Introduce a [illegible] user to select from the following o[illegible]ns

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

Extension 3

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

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

Extension 4

Modify the application to include two **Number**[illegible]**d** lists, enabling a co[illegible] Each player has their own **NumbersAll**[illegible]**d** list. On each turn, each playe[illegible] which can only use values fro[illegible]n list. This will evaluate to two oper[illegible] then enter a third expr[illegible]n which uses these two operands to identify a t[illegible] together to [illegible]ify t[illegible]gets.

Extension 5

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

Preview of Questions Ends Here

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 divisi n by zero errors, allow program to display an error message and request a new input . . . c crashing. [1]
12	(a)	The **GameOver** variable is set to True when the first ng in **Targets** list is no longer a `Targets[0] != -1`). [1]
	(b)	It prevents the loop from running indefin ly ing that the game ends when all relevant conditions have been met. [1]
13		Any 2 from: • The highest e ou e stored in a file or a database. [1] • e s game, the file/database would be read to retrieve the previous high s • ead game, if the new score exceeds the old high score, the file/database would e new value. [1]
14	(a)	**Cr eTargets / FillNumbers / ConvertToRPN / RemoveNumberUsed / UpdateTargets**
	(b)	**TrainingGame** [1]
	(c)	**UserInput, Number** [1]
	(d)	RemoveAt / 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 e of the operator on t **Operators** stack. [1] While it is greater, the top of the stack is y p ed onto **UserInputInRPN** output. [A final single check is carried o whether the top of the stack has the same pre current operator. If it h ac is popped once more onto the **UserInputInRPN** output
18		Decomposition: ra s broken into smaller tasks, each handled by specific functio Ab on: ated by hiding the complexity of certain tasks behind clear, high-level p ec roles. [1]
19		C elements in the target list, n operations will be carried out. [1]

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 [illegible] 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 evalu
- Iterating through the NumbersAllowed list to test [illegible]. [1 mark]
- Rotating the NumbersAllowed list (or similar functionality) to test different permutations of numbers.
- Appropriately displaying the combination of text expressions and associated evaluations on the scre
- Use of recursion to try combinations [1 mark]
- Only storing suitable solutions for targets which have not already been identified. [1 mark]
- Correctly [illegible] expressions which use division to ensure they evaluate to an integer. [1 mark]
- Testing ex[illegible] to ensure they correctly follow BIDMAS if needed (required for expressions built
- 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 ma

Because the expression is built up step by step, it must be tested at each stage because the impact of BI

Example Solution

Modification of the `PlayGame` method:

```
        While Not GameOver
            DisplayState(Targets, NumbersAllowed, Score)
            'CHANGE
            Console.WriteLine("Would you like helper suggesti[illegible]Y/N")
            Dim UserChoice As String = Console.ReadLine([illegible])
            If UserChoice = "Y" Then
                Dim Temp As New List(Of Integer)
                Dim PossibleSolutions As List(Of Dictionary(Of Integer, String))()
                For Each Item As Integer In NumbersAllowed
                    Temp.Add([illegible])
                Next
                For Integer = 0 To 4
                    Dim TestSolutions As Dictionary(Of Integer, String) = GenerateE
                    For Each Solution As KeyValuePair(Of Integer, String) In TestSo
                        If Not PossibleSolutions.ContainsKey(Solution.Key) Then
                            PossibleSolutions.Add(Solution.Key, Solution.Value)
                        End If
                    Next
                    Temp.Add(Temp(0))
```

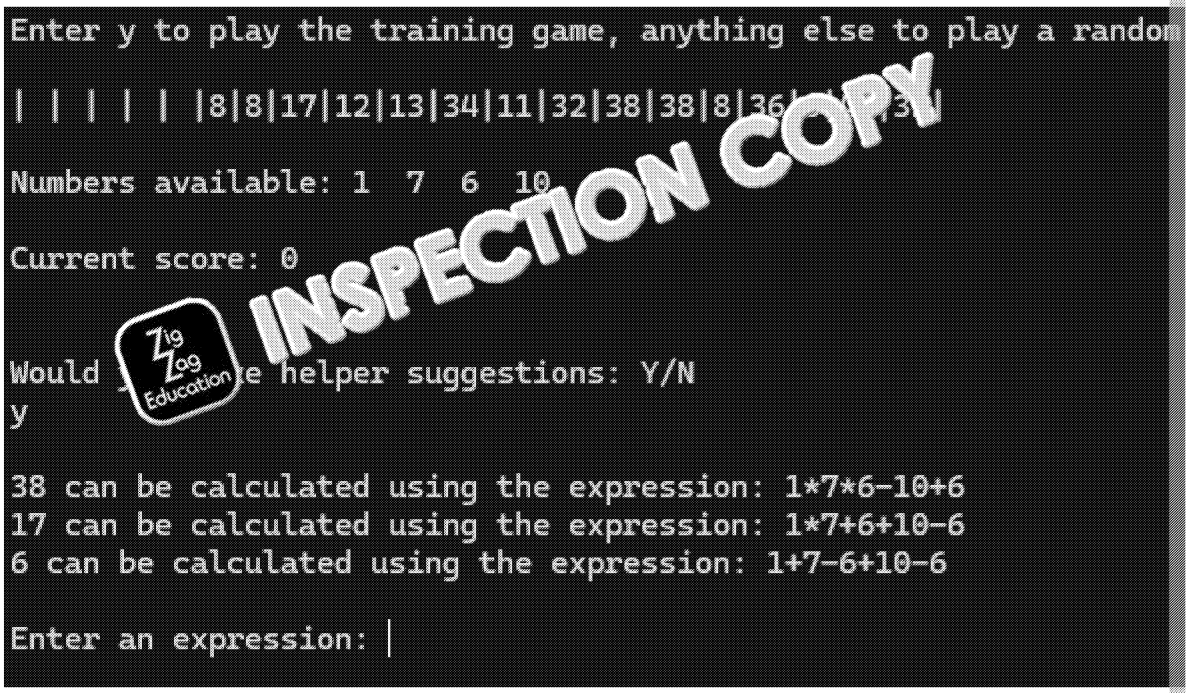
```
                    Temp.RemoveAt(0)
                Next
                Console.WriteLine()
                For Each Solution As KeyValuePair(Of Integer, String) In PossibleSo
                    Console.WriteLine($"{Solution.Key} can be calculated using the
                Next
                Console.WriteLine()
            End If
            'END CHANGE
```

Creation of new GenerateEvaluations method (and associated helper method):

```
    'CHANGE
    Public Function Gener  valuations(NumbersAllowed As List(Of Integer), Targets A
        Dim Po leE  ions As New Dictionary(Of Integer, String)()
        Gen luationsHelper(NumbersAllowed, Targets, 0, NumbersAllowed(0), Po
        Retu bleExpressions
    End Funct

    Private Sub GenerateEvaluationsHelper(NumbersAllowed As List(Of Integer), Targe
CurrentResult As Integer, PossibleExpressions As Dictionary(Of Integer, String), Cu
        If Index = NumbersAllowed.Count - 1 Then
            'Because the recursion calculates expressions step by step rather than
            'the new code needs to test the end result using RPN evaluator to ensur
            If Targets.Contains(EvaluateRPN(ConvertToRPN(CurrentExpression))) AndAl
1 Then
                If Not PossibleExpressions.ContainsKey(EvaluateRPN(ConvertToRPN(Cur
                    PossibleExpressions.Add(EvaluateRPN(ConvertToRPN(CurrentExpress
                End If
            End If
            Return
        End If

        Dim NextNumber As Integer = NumbersAllowed(In
        GenerateEvaluationsHelper(NumbersAllowed, Ta gets, ndex + 1, CurrentResult
$"{CurrentExpression}*{NextNumber}")
        If NextNumber <> 0 Then
            If (Convert.ToDoubl rentR ult / NextNumber) - Math.Truncate(Conver
                GenerateEv  s  per(NumbersAllowed, Targets, Index + 1, CInt(
$"{CurrentExpression} {NextNumber}")

        End
        Gene  uationsHelper(NumbersAllowed, Targets, Index + 1, CurrentResult
$"{CurrentExp ssion}+{NextNumber}")
        GenerateEvaluationsHelper(NumbersAllowed, Targets, Index + 1, CurrentResult
$"{CurrentExpression}-{NextNumber}")
    End Sub
    'END CHANGE
```

Testing

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



Preview of Answers Ends Here

Name	

ZigZag Education supporting

A Level AQA Computer Science Pap

Summer 2025

TARGET CLEAR

A LEVEL AQA

PRE-RELEASE 20

Electr 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	