

2015 specification
for the 2025 exam



PAPER 1 EXAM RESOURCE PACK 2025

for A Level AQA Computer Science

C# EDITION

- DIGITAL RESOURCE -

This pack includes paper versions of the electronic files.

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



POD
12389

zigzageducation.co.uk

Publish your own work... Write to a brief...
Register at publishmenow.co.uk

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

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 (56 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.

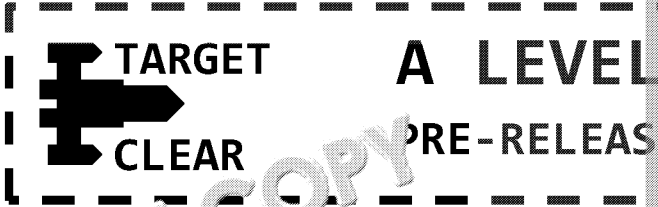
Passwords: All of the PDFs accessible via the *Solutions* web page are password-protected, so that students can only access them with your permission. Each password is a four-digit code, as follows:

- c02a-UML-Diagram-Complete.pdf
- c06-TheoryQuestions-MS.pdf
- c07-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
CheckIfUserInputEvaluationIsATarget		
Parameters	Targets : Integer List UserInputInRPN : 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 to evaluate the user inputted expression, UserInputEvaluation . The method then sets the UserInputEvaluation parameter to have a default of false. The method tests if the UserInputEvaluation parameter could not be evaluated. If the UserInputEvaluation parameter could not be evaluated, the method performs a count-controlled loop through the Targets list. The loop compares the UserInputEvaluation parameter with each target. If a match is found the Score is incremented by 1 and the UserInputEvaluation parameter is set to true. Once the loop is complete, the current score is returned.
Return values	UserInputEvaluationIsATarget : Bool	
CheckIfUserInputValid		
Parameters	UserInput : String	This method uses a Regular Expression to validate the infix expression. The Regular Expression used is: <code>^[0-9+\-*/\(\)\.]*</code> To match, the UserInput parameter must contain only mathematical operators which can only be used as literal characters. This entire expression must be repeated one or many times. The string must end with a space character. If the UserInput parameter matches the Regular Expression, the method returns true, otherwise it returns false.
Return values	Bool	

INSPECTION COPY

COPYRIGHT PROTECTED



CheckNumbersUsedAreAllInNumbersAllowed		
Parameters	NumbersAllowed : Integer List UserInputInRPN : String List MaxNumber : Int	<p>This method is used to test if the number</p> <p>The method firstly creates a temporary list from the NumbersAllowed list assigning copies of the values. The values are then passed as references not values, by using the List class. This prevents multiple references to the same list. The method removed values directly from the application elsewhere.</p> <p>The method then iterates through the UserInputInRPN list using the CheckValidNumber to confirm the elements are valid. It ensures that only operands are compared. It subsequently checks if the operand is contained in the Temp list. If the operand is NOT in the Temp list because it has found an operand which is not valid.</p> <p>The CheckValidNumber check does not check if the UserInputInRPN does not meet with the MaxNumber, the method returns false.</p>
Return values	Bool	



INSPECTION COPY

CheckValidNumber		
Parameters	Item : String MaxNumber : Int	<p>This method checks if a value passed to the game.</p> <p>This method uses a Regular Expression to check if the value is an integer number.</p> <p>The Regular Expression used is: <code>^[0-9]+</code></p> <p>To match the value parameter must be a Regular Expression pattern, the method returns true if the value is an integer. The method then tests if the value is equal to the MaxNumber parameter. If the value is not equal to the MaxNumber parameter, the method returns false.</p>
Return values	Bool	



INSPECTION COPY

INSPECTION COPY

COPYRIGHT
PROTECTED



ConvertToRPN	
Parameters	UserInput : String
Return values	UserInputInRPN: String List
<p>This method converts the infix expression to postfix notation using a version of the shunting yard algorithm.</p> <p>Initialises the following local variables:</p> <ul style="list-style-type: none"> • Position to 0. This is used to identify the current character in the expression. • Precedence to Dictionary of type <code><string, int></code> with an associated value. Multiplication and Division are given a higher precedence than Addition and Subtraction. This is used to allow the algorithm to recognise Brackets or Indices. • Operand as an integer. This uses the value of the current number in the infix notation. • UserInputInRPN as a list of strings. This is used to store the postfix expression casted as a string. • Operators as a list of strings. This is used to store the operators in the UserInput expression. <p>The method then enters a condition-compile loop. The Operand is updated using the GetNumber method. The Position variable is passed to this variable within that method as it iterates through the expression. The updated Operand is appended to the expression (assuming it is valid) multiple times to form the postfix expression.</p> <p>If the Position variable is less than the length of the expression which have not yet been processed, the method has just extracted an operand from the expression. The CurrentOperator is then incremented in the Operators list by position. The Precedence dictionary is used to compare their worth. If the value of the CurrentOperator is less than the CurrentOperator, it is added to the Operators list. The CurrentOperator is then added to the Operators list. Division functions are added to the UserInputInRPN list.</p> <p>If the Position variable is not less than the length of the expression, the operators from the string have been extracted. The CurrentOperator is then popping values from the back of the list. The method then returns the completed postfix expression.</p>	



INSPECTION COPY

COPYRIGHT
PROTECTED



CreateTargets		
Parameters	SizeOfTargets : Int MaxTarget : Int	This method populates the Targets list
Return values	Targets : Integer List	The method initialises the Targets integer list with five indices with the value -1.
		It then uses a second count-controlled loop to continue populating the list with values. In a standard pre-release game this will result in a list of 50 targets.
DisplayNumbersAllowed		
Parameters	NumbersAllowed : Integer List	This method is used to display all the values in the NumbersAllowed list.
Return values		The method iterates through the NumbersAllowed list and appends an f string to a StringBuilder.
DisplayScore		
Parameters	Score : Int	This method displays the current game score.
Return values	n/a	
DisplayState		
Parameters	Targets : Integer List NumbersAllowed : Integer List Score : Int	This method displays the current state of the game.
Return values	n/a	<ul style="list-style-type: none"> • DisplayTargets – to display the current targets • DisplayNumbersAllowed – to display the numbers allowed • DisplayScore – to display the current score
DisplayTargets		
Parameters	Targets : Integer List	This method is used to display all the values in the Targets list.
Return values	n/a	The method iterates through the Targets list and appends a pipe symbol to a StringBuilder. The method also appends a blank space onto the screen, otherwise the targets will be displayed on the same line.



INSPECTION COPY

INSPECTION COPY

COPYRIGHT
PROTECTED



EvaluateRPN		
Parameters	UserInputInRPN : String List	This method evaluates the RPN version of the expression. The expression evaluates to an integer (positive or negative). This method initializes a string list S . The controller iterates through the UserInputInRPN list. The method iterates through the UserInputInRPN list and adding elements which are not in S from UserInputInRPN and pushing their values from the start of the post-fix list, the loop stops, and the result is stored in Num2 and Num1 (essential for the next evaluation). This process is repeated until the UserInputInRPN list has been evaluated and the list S only now contains the result. The method then subtracts a truncated value from Num2. The result evaluates to 0.0, then the result must have been evaluated to a decimal and therefore can be cast as an integer.
Return values	Int	
FillNumbers		
Parameters	NumbersAllowed : Integer List TrainingGame : Bool MaxNumber : Int	This method regulates the NumbersAllowed list. If the TrainingGame parameter is true, a pre-generated list with the values 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100 is used. If the TrainingGame parameter is false, a condition-controlled loop to append values to get a new in-range target until the list is full.
Return values	NumbersAllowed : Integer List	
GetNumber		
Parameters	Number : Int	This method returns a random number between 1 and the value of the Number parameter.
Return values	Int	



INSPECTION COPY



INSPECTION COPY

INSPECTION COPY

**COPYRIGHT
PROTECTED**



GetNumberFromUserInput		
Parameters	UserInput : String Position : Int	This method is used to extract numbers converted into postfix.
Return values	Int	The method initializes / instantiates an empty string. The method iterates through the UserInput parameter to set the index of the reference rather than by value, therefore finishes. Each character is checked using the %c operator. If it is a digit (0-9), it is concatenated onto the Number variable. If it is an operator, it must be an operator which is not a digit. The loop also exits if the Position variable is iterated to the end of the string.
		If the Number variable is an empty string, the method returns -1. If the Number variable is not empty, it returns the value of the Number variable.
GetTarget		
Parameters	MaxTarget : Int	This method returns a random number between 0 and MaxTarget.
Return values	Int	
GetNumber		
Parameters	MaxNumber : Int	This method returns a random number between 0 and MaxNumber.
Return values	Int	



INSPECTION COPY



INSPECTION COPY

INSPECTION COPY

COPYRIGHT
PROTECTED



Main	
Parameters	default
Return values	n/a
<p>This is the main entrance point for the application. It allows the user to use a standard game with a randomly generated content list or a training game with fixed content lists.</p> <p>It initializes the following variables with default values:</p> <ul style="list-style-type: none"> • <code>NumberAllowed</code> as an integer list. • <code>Targets</code> as an integer list. • <code>MaxNumberOfTargets</code> as an integer. • <code>MaxTarget</code> as an integer. • <code>MaxNumber</code> as an integer. • <code>TrainingGame</code> as a Boolean. <p>The method asks the user if they would like to play a training game. If the user selects a training game, then the following variables are set in the game:</p> <ul style="list-style-type: none"> • <code>MaxTarget</code> = 1000 • <code>MaxNumber</code> = 1000 • <code>TrainingGame</code> = true • The <code>Targets</code> list is populated with 20 random integers between 1 and 1000. <p>If the user does not select a training game, then the following variables are set for use later in the game:</p> <ul style="list-style-type: none"> • <code>MaxTarget</code> = 10 • <code>MaxNumber</code> = 50 • <code>TrainingGame</code> = false • The <code>Targets</code> list is populated with 20 random integers between 1 and <code>MaxTarget</code> inclusive. <p>The method then calls the <code>FillNumbers</code> method to populate the <code>NumberAllowed</code> list and the <code>PlayGame</code> 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"> • Score to 0 • GameOver to false. • UserInput as a string. • UserInputInRPN as a list of strings These variables are then used and populated.
Return values	n/a	The method then enters into the main algorithm. <ul style="list-style-type: none"> • Call the DisplayState method pass the Score and Targets to display the current values in these variables. • Prompt the user to enter an infix mathematical expression into the UserInput variable. • Call the CheckIfUserInputValid method to check if the input is a valid infix expression. • If the input is valid, the ConvertToRPN method is called, which converts the infix UserInput into reverse Polish notation and stores it in the UserInputInRPN list. • Call the CheckNumbersUsedAreValid method to check if the NumbersAllowed list, UserInputInRPN list and the Score are valid. • If all the values in the UserInputInRPN list are valid, the CheckIfUserInputEvaluationIsATarget method is called, passing in the UserInputInRPN list and the Score variable. This method evaluates the UserInputInRPN list and the Score variable to see if the UserInputInRPN list evaluates to one of the Targets rather than as a value. • If UserInputInRPN evaluates to one of the Targets, the Score variable is appropriately incremented. The UpdateScore method is then called, passing in the Score variable, MaxNumber variable and the Targets list to update the Score variable and the Targets list. The UpdateScore method is then called, passing in the Score variable, MaxNumber variable and the Targets list to update the Score variable and the Targets list. • The Score variable is then decremented by one if the UserInputInRPN list successfully identified a target. • The method then tests to see if the GameOver variable is set to true with the CheckIfGameOver method. If the GameOver variable is set to true with the CheckIfGameOver method, the UpdateScore method is called, passing in the Score variable, MaxNumber variable and the Targets list together with the TrainingGame array to update the Score variable and the Targets list one index to the left. If the GameOver variable has been set to true, the UpdateScore method is called, passing in the Score variable, MaxNumber variable and the Targets list together with the TrainingGame array to update the Score variable and the Targets list one index to the left. The final Score and the GameOver variable are then displayed.



INSPECTION COPY

COPYRIGHT
PROTECTED



RemoveNumbersUsed		
Parameters	UserInput : String MaxNumber : Int NumbersAllowed : Integer List	This method removes any numbers from an evaluation match with a target.
Return values	n/a	The method first calls the ConvertToRPN method to convert the user input to a Reverse Polish Notation (RPN) expression. Although when using the ConvertToRPN method the UserInputInRPN parameter is passed as a reference, the UserInputInRPN parameter is passed as a value by default, passed as references not by value. The method then calls the CheckIfUserInputEvaluationIsATarget method to check if the UserInputInRPN list, consequently ReturnValidNumber expression from the user to rebuild a new expression.
		The method then iterates through the UserInputInRPN list and calls the CheckValidNumber method to confirm the element is a valid number to ensure that only operands are compared. The method then checks if the operand is contained in the NumbersAllowed list.
UpdateTargets		
Parameters	Targets : Integer List TrainingGame : Bool MaxTarget : Int	This method uses a count-controlled loop to update the list with a new value. This results in the list being backfilled with a new value. This results in the list being backfilled with a new value.
Return values	n/a	The method firstly iterates through the list and updates the value. This has the effect of moving each value to the next position in the list. The method then removes the last element from the list. The method then uses selection on the list to find the maximum value. The method then uses selection on the list to find the maximum value and therefore the value at the end of the list. If false, the user has not passed the target parameter MaxTarget. The method then adds the value (inclusive) and adds it to the list.



INSPECTION COPY

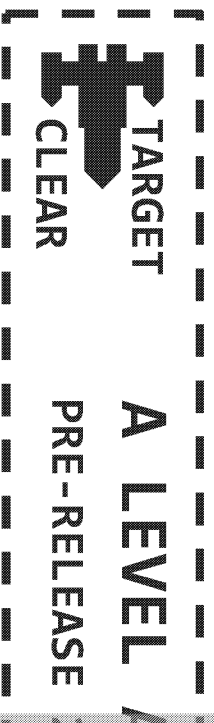


INSPECTION COPY

INSPECTION COPY

COPYRIGHT PROTECTED





Training Game Expression

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

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

- 68 = $(8+2)/3+2+2$
- 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 = $(8-3) * \text{Log}_2 512$
- 43 = (Concatenate 2 and $\text{Log}_8 512) *$

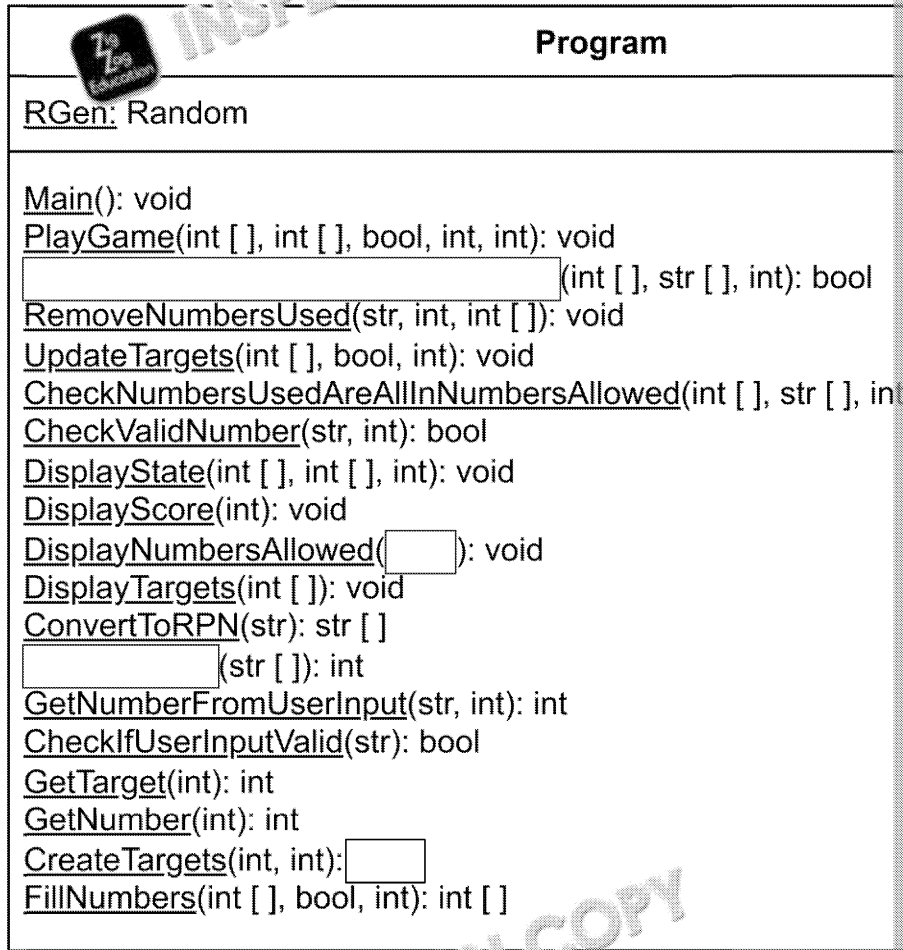
INSPECTION COPY

INSPECTION COPY



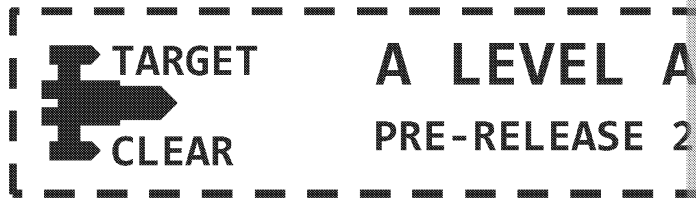
COPYRIGHT
PROTECTED

UML Class Diagram



COPYRIGHT
 PROTECTED





INSPECTION COPY

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

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

.....

.....

.....

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

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

.....

.....

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

.....

.....

.....

**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

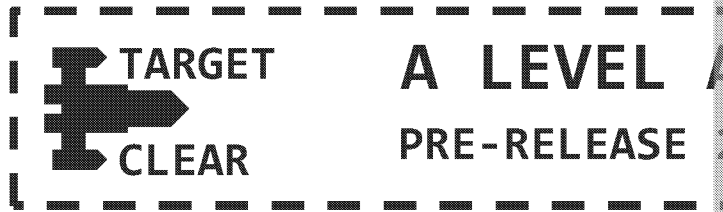
END OF QUESTIONS

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:

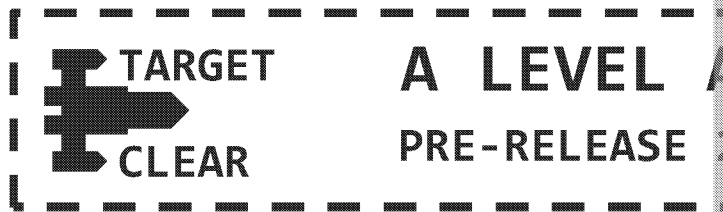
- Your PROGRAM SOURCE CODE showing the modifications to the `PlayGame` method.
- SCREEN CAPTURE(S) showing the required tests.

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 different score to be awarded depending on how close the user's calculation is to a target. Award 1 point if the user's calculation is exactly equal to the target. Award 3 points 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	<p>(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]</p> <p>(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]</p>
12	<p>(a) The GameOver variable is set to true when the first target in the Targets list is no longer available. Targets[0] != -1. [1]</p> <p>(b) It prevents the loop from running indefinitely, ensuring that the game ends when all relevant conditions have been met. [1]</p>
13	<p>Any 2 from:</p> <ul style="list-style-type: none"> The highest score would be stored in a file or a database. [1] At the start of each game, the file/database would be read to retrieve the previous high score. [1] 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]
14	<p>(a) CreateTargets / FillNumbers / ConvertToRPN / RemoveNumberUsed / UpdateTargets [1]</p> <p>(b) TrainingGame [1]</p> <p>(c) UserInput, Number [1]</p> <p>(d) RemoveAt / Add [1]</p> <p>(e) MaxTarget / MaxNumber / MaxNumberOfTargets [1]</p>
15	<p>Any 2 from:</p> <ul style="list-style-type: none"> + - 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) followed by an operator [1]
16	<p>Because regular expressions do not support recursion. [1]</p> <p>A regular expression cannot track the opening and closing of brackets / a regular expression cannot maintain a "state". [1]</p>
17	<p>The precedence of the current operator is compared to the precedence of the operator on top of the Operators stack. [1]</p> <p>While it is greater, the top of the stack is popped and the result is added to the UserInputInRPN output. [1]</p> <p>A final single check is carried out to ensure whether the top of the stack has the same precedence as the current operator. If it has, it is popped once more onto the UserInputInRPN output. [1]</p>
18	<p>Decomposition: The problem is broken into smaller tasks, each handled by specific functions. [1]</p> <p>Abstraction: The complexity of certain tasks is hidden behind clear, high-level functions. [1]</p>
19	<p>Each element in the target list, n operations will be carried out. [1]</p>

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 use helper suggestions. [1 mark]
- 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. [1 mark]
- Iterating through the NumbersAllowed list to test permutations. [1 mark]
- Rotating the NumbersAllowed list (or similar) to test different permutations of numbers. [1 mark]
- Appropriately displaying the combination of text expressions and associated evaluations on the screen. [1 mark]
- 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). [1 mark]
- 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 should be awarded for recursion.

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

Example Solution

Modification of the PlayGame method:

```
while (!GameOver)
{
    DisplayState(Targets, NumbersAllowed, Score);
    //CHANGE
    Console.WriteLine("Would you like helper suggestions: Y/N");
    string UserChoice = Console.ReadLine().ToUpper();
    if (UserChoice == "Y")
    {
        List<int> Temp = new List<int>();
        Dictionary<int, string> PossibleSolutions = new Dictionary<int, string>();
        foreach (int Item in NumbersAllowed)
        {
            Temp.Add(Item);
        }
        for (int i = 0; i < 5; i++)
        {
            Dictionary<int, string> TestSolutions = GenerateEvaluations(Temp, PossibleSolutions);
            foreach (KeyValuePair<int, string> Solution in TestSolutions)
            {
                if (Solution.Key == Target)
                {
                    Console.WriteLine("Solution found: " + Solution.Value);
                    return true;
                }
            }
        }
    }
}
```

INSPECTION COPY

COPYRIGHT
PROTECTED





```

        if (!PossibleSolutions.ContainsKey(Solution.Key))
        {
            PossibleSolutions.Add(Solution.Key, Solution.Value)
        }
    }
    Temp.Add(Temp[0]);
    Temp.RemoveAt(0);
}
Console.WriteLine();
foreach (KeyValuePair<int, string> solution in PossibleSolutions)
{
    Console.WriteLine("{Solution.Key} can be calculated using");
}
Console.WriteLine();
//CHANGE
Console.WriteLine("Enter an expression: ");
UserInput = Console.ReadLine();

```

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

```

//CHANGE
public static Dictionary<int, string> GenerateEvaluations(List<int> NumbersAllowed, int Target)
{
    Dictionary<int, string> PossibleExpressions = new Dictionary<int, string>();
    GenerateEvaluationsHelper(NumbersAllowed, Targets, 0, NumbersAllowed[0], Target, PossibleExpressions, NumbersAllowed[0].ToString());
    return PossibleExpressions;
}

private static void GenerateEvaluationsHelper(List<int> NumbersAllowed, List<int> Targets, int Index, Dictionary<int, string> PossibleExpressions, string CurrentExpression)
{
    if (Index == NumbersAllowed.Count - 1)
    {
        //Because the recursive method creates expressions step by step rather than all at once
        //the new code is needed to test the end result using RPN evaluator to see if it matches the target
        if (Target == EvaluateRPN(EvaluateRPN(ConvertToRPN(CurrentExpression))))
        {
            PossibleExpressions.Add(CurrentExpression, CurrentExpression);
        }
    }
    return;
}

```



INSPECTION COPY

**COPYRIGHT
PROTECTED**




```

        int NextNumber = NumbersAllowed[Index + 1];
        GenerateEvaluationsHelper(NumbersAllowed, Targets, Index + 1, CurrentRes
        $"{CurrentExpression}*{NextNumber}");
        if (NextNumber != 0)
        {
            if ((double)CurrentResult / NextNumber - Math.Truncate((double)Curr
            {
                GenerateEvaluationsHelper(NumbersAllowed, Targets, Index + 1, C
        $"{CurrentExpression}/{NextNumber}");
            }
        }
        GenerateEvaluationsHelper(NumbersAllowed, Targets, Index + 1, CurrentRes
        $"{CurrentExpression}+{NextNumber}");
        GenerateEvaluationsHelper(NumbersAllowed, Targets, Index + 1, CurrentRes
        $"{CurrentExpression}-{NextNumber}");
    }
    //END

```

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 suggestion
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

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

