****

**Programming Tasks**

These questions require you to load the **Skeleton Program** and to make programming changes to it.

*Note that any alternative or additional code changes that are deemed appropriate to make must also be evidenced, ensuring that it is clear where in the Skeleton Program those changes have been made.*

*The objective of this resource is to provide you with a selection of different questions and solutions to those questions. Some questions are more prescriptive than others in how the task should be completed to support a range of learners. Questions which have a similar theme may use different techniques to give students a range of options on how to solve problems. Some Regular Expression solutions use meta characters which are arguably beyond the AQA 7517 specification but make the solution considerably simpler. Students are encouraged to learn 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 before attempting each of the questions in this resource. This will prevent modifications made for one question having an unintended impact on a different question.*

# Task 1

**Task 1** **Marks:** 3

This question extends the Skeleton Program to allow the user to end the game at any point rather than wait 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 quit and display the 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 1 point.
* 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. [2 marks]
* SCREEN CAPTURE(S) showing the required tests. [1 mark]

**Task 2** **Marks:** 6

This question extends the Skeleton Program by including suitable error messages to the user. The application does not give any feedback to the user for any invalid or erroneous inputs. Modify the PlayGame, CheckNumbersUsedAreAllInNumbersAllowed and CheckIfUserInputValid methods to give suitable feedback to the user in the following conditions:

* The expression they enter does not evaluate to a target in the Targets list.
* The expression they enter does not include valid numbers from the NumbersAllowed list (including numbers which are greater than MaxNumber).
* The expression they enter is not valid infix notation.
* The expression includes division by zero. Use a Regular Expression in the CheckIfUserInputValid to detect this.

**What you need to do**

**Task 2.1**

Update the PlayGame, CheckNumbersUsedAreAllInNumbersAllowed and CheckIfUserInputValid methods to give a suitable error message to the user in the way described.

**Task 2.2**

Test that the changes you have made work:

* Run the Skeleton Program.
* Enter y to start a training game.
* Show the program displaying a suitable error message for the expression: 512\*2
* Show the program displaying a suitable error message for the expression: 8+14
* Show the program displaying a suitable error message for the expression: 4608/512
* Show the program displaying a suitable error message for the expression: 8/0

**Evidence that you need to provide:**

* Your PROGRAM SOURCE CODE showing the introduction of a new error handling code. [5 marks]
* SCREEN CAPTURE(S) showing the required tests. [1 mark]

**Task 3** **Marks:** 2

The application currently uses list data structures but includes some functionality which would be better suited to stack or queue data structures.

The UpdateTargets method uses iteration on a list to interact with the list as if it were a queue. Because the iteration is upperbound by the value of the MaxNumbersOfTargets variable, it will always iterate the same number of times. The method therefore has a time complexity of O(1). The code, however, could be simplified to just interact with the start and end of the queue rather than iterating through it. This would make the functionality cleaner.

**What you need to do**

**Task 3.1**

Modify the UpdateTargets method to simplify how the list is interacted just using start and end indices rather than iterating through it. Ensure that the functionality of the method does not change.

**Task 3.2**

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 displaying the updated Targets list with the targets moved one index to the left and 119 added to the end of the list.

**Evidence that you need to provide:**

* Your PROGRAM SOURCE CODE showing the refactoring of the UpdateTargets method. [1 mark]
* SCREEN CAPTURE(S) showing the required tests. [1 mark]

**Task 4** **Marks:** 8

This question extends the functionality of the Skeleton Program to introduce difficulty levels to the game.

Introduce the functionality to have “Easy”, “Medium” and “Hard” levels in the standard random game. Introduce a list of “Large” numbers [ 25, 50, 75, 100 ]. The application should populate the NumbersAllowed list as follows:

|  |  |  |
| --- | --- | --- |
| **Game Mode** | **How many “Large” numbers** | **How many standard random numbers** |
| Standard | 0 | 5 |
| Easy | 1 | 4 |
| Medium | 2 | 3 |
| Hard | 4 | 1 |

**The functionality should produce a brand-new set of NumbersAllowed for each turn even if the user expression does not evaluate to a target.**

*This functionality is not applicable to a training game.*

**What you need to do**

**Task 4.1**

Modify the Main method to introduce a menu to give the user a choice of the type of standard random game that they would like to play.

Modify the FillNumbers method and any other methods required to introduce a difficulty level option in the way described.

**Task 4.2**

Test that the changes you have made work:

* Run the Skeleton Program.
* Press enter to start a standard game.
* Show the program displaying a suitable menu and prompting the user.
* Select a “medium” difficulty game.
* Show the program displaying a NumbersAllowed list which contains two values from the list

[ 25, 50, 75, 100 ].

**Evidence that you need to provide:**

* Your PROGRAM SOURCE CODE showing the amended Main and FillNumbers   
  methods and any other methods you have modified when answering this question. [7 marks]
* SCREEN CAPTURE(S) showing the required tests. [1 mark]

**Task 5** **Marks:** 4

This question is about the use of the values in the NumbersAllowed list. The application currently awards the same number of points regardless of the complexity of the expression entered by the user. This does not encourage mathematical extension and challenge.

Introduce functionality into the application to give a bonus of 2 points for every operand used in the expression entered, if the evaluation is a valid target.

**What you need to do**

**Task 5.1**

Modify the CheckIfUserInputEvaluationIsATarget method to operate as described.

**Task 5.2**

Test that the changes you have made work:

* Run the Skeleton Program.
* Enter y to start a training game.
* Enter the expression: 512/8+2+2
* Show the program correctly identifying the target 68 and increasing the user score to 9 points (8 additional points for four operands used in the expression).

**Evidence that you need to provide:**

* Your PROGRAM SOURCE CODE showing the amended CheckIfUserInputEvaluationIsATarget method. [3 marks]
* SCREEN CAPTURE(S) showing the required tests. [1 mark]

**Task 6** **Marks:** 4

This question extends the Skeleton Program by allowing the user to move the Targets list back to the right, but at a cost to their score.

New functionality should be introduced which offers the user the opportunity to move the targets back to the right instead of entering an expression. If they select this option, the Targets list should be moved along to the right by one index and the Score should be reduced by 2. The list should maintain its standard length. The start of the list should be repopulated with -1.

**What you need to do**

**Task 6.1**

Modify the PlayGame method to give the user the option to enter an expression or enter “MOVE” to move the Targets list.

Create a new method called MoveTargetsBack which operates as described.

**Task 6.2**

Test that the changes you have made work:

* Run the Skeleton Program.
* Enter y to start a training game.
* Enter the expression: 512/8+2+2
* Enter the expression: 8+3-2
* When prompted to enter another expression, select to move the Targets list to the right.
* Show the program displaying the Targets list with the order:

| | | | |23| |140|82|121|34|45| |75|34|23|119|43|23|119|119|

**Evidence that you need to provide:**

* Your PROGRAM SOURCE CODE showing the new method MoveTargetsBack and   
  the amended PlayGame method. [3 marks]
* SCREEN CAPTURE(S) showing the required tests. [1 mark]

**Task 7** **Marks:** 7

This question extends the Skeleton Program to allow the user to use correctly identified targets in   
future expressions.

For example, in a standard random game with a Targets list and a NumbersAllowed list of:

| | | | | |43|46|24|42|3|15|27|35|8|28|5|4|3|35|36|

Numbers available: 1 9 6 8 2

If the user enters the expression 9+6, this correctly identifies the target number 15. The user should then be given the option to either add the number 15 or just 1 or just 5 into the NumbersAllowed list to be used in a later turn. The NumbersAllowed list can increase to more than five values. This functionality can add a 0 to the NumbersAllowed list if the identified target is a single digit or is divisible by 10.

Introduce new functionality, so that when a target is correctly identified, give the user the option to add the target or part of it into their NumbersAllowed list. The program should display the target and its component digits to the user to allow them to choose. If they select to add the target (or one of its component digits) it should be added to the NumbersAllowed list.

**What you need to do**

**Task 7.1**

Modify the PlayGame and CheckIfUserInputEvaluationIsATarget methods to display when a target has been correctly identified and prompt the user to choose if they would like to use it to add into the NumbersAllowed list.

Create a new method called SelectValueFromTarget which displays the correctly identified target and its component digits and invites the user to select which part they would like to add to the NumbersAllowed list.

**Task 7.2**

Modify the FillNumbers method to add the selected target from the user to the NumbersAllowed list whilst maintaining its functionality at the start of the game of initially populating the NumbersAllowed list.

**Task 7.3**

Test that the changes you have made work:

* Run the Skeleton Program.
* Press enter to start a standard random game.
* Enter an expression which correctly identifies a target greater than 9.
* Show the program prompting the user to choose if they would like to use the target in their NumbersAllowed list. Select this option.
* Show the program prompting the user which digit in the target they would like to use. The program should show the target and both digits individually. Select one of the options available.
* Show the program correctly adding the selected option to the NumbersAllowed list.

**Evidence that you need to provide:**

* Your PROGRAM SOURCE CODE showing the amended CheckIfUserInputEvaluationIsATarget and PlayGame methods. [2 marks]
* Your PROGRAM SOURCE CODE showing the new SelectValueFromTarget method. [2 marks]
* Your PROGRAM SOURCE CODE showing the amended FillNumbers method. [2 marks]
* SCREEN CAPTURE(S) showing the required tests. [1 mark]

**Task 8** **Marks:** 8

This question extends the Skeleton Program by introducing a challenge mode into the program awarding the player a bonus score if they use all the numbers available to them and deduct points if they don’t

Introduce a “Challenge Mode” to the application. There should be a 25% chance of the “Challenge Mode” being triggered each turn. When triggered, the user should be told that they need to use all five numbers from the NumbersAllowed list in that turn. The user should be told that doing so will gain them a bonus of 10 points and that if they do not, they will be deducted 5 points. If an expression identifies multiple targets the user should only be awarded or deducted points once.

**What you need to do**

**Task 8.1**

Modify the PlayGame method to tell the user that the “Challenge Mode” has been activated. There should be a 25% chance of this happening each turn.

Modify the CheckIfUserInputEvaluationIsATarget in Challenge Mode so that if a target is correctly identified and the user has used all the numbers in the NumbersAllowed list, the user is awarded an additional 10 points. If a target is identified but the user has not used all the numbers in the NumbersAllowed list, the user score should be reduced by 5 points.

**Task 8.2**

Test that the changes you have made work:

* Run the Skeleton Program.
* Press enter to start a training game.
* Enter expressions until the “Challenge Mode” is activated.
* Show the program telling the user that “Challenge Mode” has been activated.
* Enter an expression which uses all five numbers in the NumbersAllowed list.
* Show the program displaying the score increasing by 10 points on top of the normal score awarded for the target(s) identified.

**Evidence that you need to provide:**

* Your PROGRAM SOURCE CODE showing the amended PlayGame method. [3 marks]
* Your PROGRAM SOURCE CODE showing the amended CheckIfUserInputEvaluationIsATarget method. [4 marks]
* SCREEN CAPTURE(S) showing the required tests. [1 mark]

**Task 9** **Marks:** 8

This question extends the Skeleton Program to allow the user to “Freeze” a number in the Targets list so that it does not move along to the left when the UpdateTargets method is called at the end of each turn.

Introduce new functionality to allow the user to select a number in the Targets list to freeze in place. The user can only have one frozen number at any time and can either freeze or unfreeze the number as part of their turn. A frozen number does not move when the number list is moved one to the left. A frozen number is shown by the symbols < > around it.

**What you need to do**

**Task 9.1**

Modify the PlayGame method to give the user the option to freeze or unfreeze a target in the Targets list and allow them to select the position. Display the state of the game again to confirm the update with the frozen number shown.

Modify the UpdateTargets method so that the number identified as frozen does not move when the Targets list is updated.

Update the DisplayState method and any subsequently required methods to display the game state   
as described.

**Task 9.2**

Test that the changes you have made work:

* Run the Skeleton Program.
* Enter y to start a training game.
* When prompted, decline the offer to freeze a number on your first turn.
* Enter the expression: 512/8+2+2
* When prompted to freeze a number, enter position 8 (which should be the target 82).
* Show the program displaying the Targets list with the order:

| | | | |23|9|140|<82>|121|34|45| |75|34|23|119|43|23|119|119|

* Enter the expression: 8+3-2
* Show the program displaying the Targets list with the order:

| | | |23| |140|121|<82>|34|45| |75|34|23|119|43|23|119|119|119|

**Evidence that you need to provide:**

* Your PROGRAM SOURCE CODE showing the amended method PlayGame and   
  the amended UpdateTargets method. [5 marks]
* Your PROGRAM SOURCE CODE showing the amended method DisplayState   
  and any subsequent methods which need updating to operate as described. [2 marks]
* SCREEN CAPTURE(S) showing the required tests. [1 mark]

**Task 10** **Marks:** 11

This question extends the Skeleton Program by introducing some object orientation to allow the user to undo previous moves.

Introduce new functionality to allow the user to undo their previous moves. If there are undo moves available, the program should inform the user how many are available and ask them if they would like to undo their last move. If they select this option, the program should undo the last move.

**What you need to do**

**Task 10.1**

Create a new class called UndoState which stores the copies of the NumbersAllowed list, Targets list and Score. Create suitable accessor methods to be able to access these properties.

Modify the PlayGame method to prompt the user appropriately and introduce a suitable data structure to store UndoState objects to fulfil the requirements described.

**Task 10.2**

Create a new method AddToUndo which instantiates a copy of the current game state and stores it accordingly to allow undo functionality.

Create a new method UndoLastTurn which restores the game to the previous game state, ensuring that the NumbersAllowed list, Targets list and Score are all updated.

**Task 10.3**

Test that the changes you have made work:

* Run the Skeleton Program.
* Enter y to start a training game.
* Enter the expression: 512/8+2+2
* When prompted, do not undo the last turn.
* Enter the expression: 8+3-2
* Show the program displaying that 2 undos are available.
* When prompted undo the last move.
* Show the program displaying the Targets list:

| | | | |23|9|140|82|121|34|45| |75|34|23|119|43|23|119|119|

**Evidence that you need to provide:**

* Your PROGRAM SOURCE CODE showing the new class UndoState with a suitable   
  constructor and accessor methods. [4 marks]
* Your PROGRAM SOURCE CODE showing the amended PlayGame method with a   
  suitable prompt as described. [2 marks]
* Your PROGRAM SOURCE CODE showing the new AddToUndo and UndoLastTurn   
  methods. [4 marks]
* SCREEN CAPTURE(S) showing the required tests. [1 mark]

**Task 11** **Marks:** 12

This question introduces a new “Help Mode” feature to the game by suggesting targets which have been identified as possible, through brute force.

Introduce a “Help Mode” to the game. Create a new method which brute forces different combinations of operators and operands from the NumbersAllowed list. Where the brute force approach finds a possible expression, which evaluates to one of the values in the Targets list, the program should highlight that target with \* \* around it. To reduce processing, the brute forcing should **only use expressions involving two or three operands, and only up to five possible targets need to be identified in each turn**. If five possible targets are not found within 30 attempts, the program should stop looking and simply display what it has found. The program **does not** need to display the actual expressions to the user. The brute forcing approach **does not** need to test every possible combination of the values in the NumbersAllowed list.

**What you need to do**

**Task 11.1**

Modify the PlayGame method to give the user the option to have up to five target suggestions identified. Use a suitable data structure to store these values.

Create a new method called **GetRandomSuggestions** which iterates through the NumbersAllowed list trying combinations of values from the list with the four mathematical operators used in this game to operate as described.

**Task 11.2**

Modify the DisplayState and DisplayTargets methods to operate as described.

**Task 11.3**

Test that the changes you have made work:

* Run the Skeleton Program.
* Press enter to start a standard random game.
* When prompted, choose to receive target suggestions.
* Show the program displaying the Targets list with suggested targets highlighted.
* Enter a suitable expression which evaluates to one of the highlighted targets.
* Show the program correctly removing the identified target and updating the score appropriately.

**Evidence that you need to provide:**

* Your PROGRAM SOURCE CODE showing the new method **GetRandomSuggestions**and the amended PlayGame method. [8 marks]
* Your PROGRAM SOURCE CODE showing the amended DisplayState and   
  DisplayTargets methods. [3 marks]
* SCREEN CAPTURE(S) showing the required tests. [1 mark]

**Task 12** **Marks:** 7

The program currently does not give a reason if an expression entered by the user does not evaluate to a target in the Targets list.

Introduce functionality to tell the user what their expression evaluated to and how many targets were found as a result. Inform the user if they have used any values which are not in the NumbersAllowed list.

**What you need to do**

**Task 12.1**

Modify the **CheckNumbersUsedAreAllInNumbersAllowed** method to tell the user if they have used any values not present in the NumbersAllowed list and which ones they are.

Modify the **CheckIfUserInputEvaluationIsATarget** method to tell the user what their expression evaluated to and how many times it was found in the Targets list.

**Task 12.2**

Test that the changes you have made work:

* Run the Skeleton Program.
* Enter y to start a training game.
* Enter the expression: 8+100
* Show the program displaying a suitable error highlighting that 100 is not valid.
* Enter the expression: 8+3-2
* Show the program displaying that 9 is a valid target and was found once in the Targets list.

**Evidence that you need to provide:**

* Your PROGRAM SOURCE CODE showing the amended **CheckNumbersUsedAreAllInNumbersAllowed** and   
  **CheckIfUserInputEvaluationIsATarget** methods. [5 marks]
* SCREEN CAPTURE(S) showing the required tests. [2 marks]

**Task 13** **Marks:** 7

This question extends the functionality of the Skeleton Program to clear the Targets list more quickly by removing all the targets between two duplicate, correctly identified, targets.

Introduce new functionality so that if an expression evaluates to **pairs of** targets, all of the numbers between those targets are also removed from the Targets list, collapsing the gap formed and achieving 2 points per target removed. The Targets list should then be backfilled accordingly depending on whether the user is playing a training game or a standard random game.

This question does NOT need to consider the scenario of more than two duplicate targets.

**What you need to do**

**Task 13.1**

Modify the **CheckIfUserInputEvaluationIsATarget** method to remove all targets from the Targets list if they are between two matching targets identified by the user. The targets themselves should also be removed to close the gap in the list. Award the user 2 points for each target removed.

Modify the **UpdateTargets** method and any other methods required to correctly repopulate the Targets list to the default number of targets.

**Task 13.2**

Test that the changes you have made work:

* Run the Skeleton Program.
* Enter y to start a training game.
* Enter the expression: 512/8/2+2
* Show the program displaying the backfilled Targets list:

| | | | |23|9|140|82|121|23|119|43|23|119|119|119|119|119|119|119|

**Evidence that you need to provide:**

* Your PROGRAM SOURCE CODE showing the amended **CheckIfUserInputEvaluationIsATarget** and **UpdateTargets** methods together with   
  updates to any other required methods. [6 marks]
* SCREEN CAPTURE(S) showing the required tests. [1 mark]

**Task 14** **Marks:** 5

This question extends the mathematical range of the Skeleton Program by introducing indices.

Introduce new functionality to all the use of positive indices in expressions. The user should be able to use the ^ operator to raise a number to the power of another number. Both values must be present in the NumbersAllowed list.

**What you need to do**

**Task 14.1**

Modify the **ConvertToRPN** method to add the new functionality with the correct precedence.

Modify the **EvaluateRPN** and CheckIfUserInputValid methods to allow the program to operate as described.

**Task 14.2**

Test that the changes you have made work:

* Run the Skeleton Program.
* Press y to start a training game.
* Enter the expression: 512/8\*2-3^2
* Show the program displaying an updated Targets list with 119 correctly removed:

| | | | |23|9|140|82|121|34|45|68|75|34|23| |43|23| | |

**Evidence that you need to provide:**

* Your PROGRAM SOURCE CODE showing the amended **ConvertToRPN** and   
  **EvaluateRPN** and CheckIfUserInputValid methods. [4 marks]
* SCREEN CAPTURE(S) showing the required tests. [1 mark]

**Task 15** **Marks:** 10

This question extends the functionality of the Skeleton Program by allowing the user to “reload” a previous game state by entering a character string which represents a game.

Introduce new functionality in the opening menu of the game to allow the user to enter L to load a previous game state using an encoded string. The user should then be prompted to enter a string of 26 characters. Continue to ask the user for a code until a valid code is entered. The first 20 characters should represent the Targets list with the @ symbol representing -1. The next five characters should represent the NumbersAllowed list, with the final character representing the Score. The functionality can only represent a standard random game because of the MaxTarget size of 50.

A screen shot of a computer screen

Description automatically generated

Subtract 64 from the ASCII equivalent of each character in the encoded game string. For example, the letter D should represent the number 4 in the Targets list or NumbersAllowed list.

**What you need to do**

**Task 15.1**

Create a new RestoreGame method to allow the user to “load” a game from a previous state encoded string. The method should give suitable error messages if the string is the incorrect length or contains invalid characters. The portion of the string representing the NumbersAllowed list and Score cannot contain the @ symbol.

Modify the **Main** and PlayGame methods to allow the program to operate as described.

**Task 15.2**

Test that the changes you have made work:

* Run the Skeleton Program.
* Enter L to load a previous game state.
* Enter the string: game.txt
* Show the program displaying a suitable error.
* Enter the string: @@@@@Qf\_Hg@kCCN@WJhSCHGJAC
* Show the program displaying the correct “loaded” game state:

| | | | | |17|38|31|8|39| |43|3|3|14| |23|10|40|19|

Numbers available: 3 8 7 10 1

Current score: 3

**Evidence that you need to provide:**

* Your PROGRAM SOURCE CODE showing the new RestoreGame method together  
  with the amended **Main** and **PlayGame** methods. [9 marks]
* SCREEN CAPTURE(S) showing the required tests. [1 mark]

**Task 16** **Marks:** 12

This question extends the mathematical range of the Skeleton Program by introducing parentheses.

Add functionality to the game to allow users to input infix notation using brackets in their expressions to control the order of operations.

You can assume that the user will enter a valid number of brackets in their expression.

**What you need to do**

**Task 16.1**

Create a new method ConvertToRPNWithBrackets to replace the ConvertToRPN method. Use a shunting algorithm to correctly interpret the precedence of an expression which includes brackets. Update the application to replace wherever the ConvertToRPN method is called.

Modify the **CheckValidOperator** and **CheckIfUserInputValid** methods to update the Regular Expressions to match with a suitable pattern to allow brackets in an infix expression.

**Task 16.2**

Test that the changes you have made work:

* Run the Skeleton Program.
* Enter y to start a training game.
* Enter the expression: (8+2)\*2+3
* Show the program displaying an updated Targets list with 23 correctly removed:

| | | | |9|140|82|121|34|45|68|75|34| |119|43| |119|119|119|

**Evidence that you need to provide:**

* Your PROGRAM SOURCE CODE showing the new ConvertToRPNWithBrackets   
  method and any additional methods required to operate. [9 marks]
* Your PROGRAM SOURCE CODE showing the amended **CheckValidOperator** and **CheckIfUserInputValid** methods. [2 marks]
* SCREEN CAPTURE(S) showing the required tests. [1 mark]

**Task 17** **Marks:** 6

This question extends the functionality of the Skeleton Program allowing the user to enter their expression using postfix notation rather than infix notation.

Add functionality to the game to allow users the option to input an expression using postfix notation or infix notation and correctly evaluate either. Expressions entered using postfix notation should use a comma as the delimiter between each element to allow elements to be correctly identified.

**What you need to do**

**Task 17.1**

Modify the PlayGame method to prompt the user if they would like to enter their expression using infix or postfix notation. If they select postfix, advise them that their expression will need to include a comma as a suitable delimiter.

Modify the ConvertToRPN method to allow it to cater for either an infix expression or a postfix expression.

**Task 17.2**

Test that the changes you have made work:

* Run the Skeleton Program.
* Enter y to start a training game.
* Select to enter an expression using RPN.
* Enter the expression: 8,2,+,2,\*,3,+
* Show the program displaying an updated Targets list with 23 correctly removed:

| | | | |9|140|82|121|34|45|68|75|34| |119|43| |119|119|119|

**Evidence that you need to provide:**

* Your PROGRAM SOURCE CODE showing the amended **PlayGame** and   
  ConvertToRPN methods. [5 marks]
* SCREEN CAPTURE(S) showing the required tests. [1 mark]

**Task 18** **Marks:** 5

The program currently allows for duplicate numbers to be generated in the Targets list and NumbersAllowed list. Removing this functionality would increase the level of challenge to the player.

Add new functionality to the game to only allow unique numbers to be added to the Targets list and NumbersAllowed list in a standard random game.

**What you need to do**

**Task 18.1**

Modify the **FillNumbers** and CreateTargets methods to allow the program to operate as described.

**Task 18.2**

Test that the changes you have made work:

* Run the Skeleton Program.
* Press enter to start a standard random game.
* Show the program displaying the NumbersAllowed list and Targets list with no duplicated values in the individual lists.
* Enter an expression with at least two operands which correctly identifies a target.
* Show the program displaying the NumbersAllowed list and Targets list with no duplicated values in the individual lists.

**Evidence that you need to provide:**

* Your PROGRAM SOURCE CODE showing the amended **FillNumbers** and   
  CreateTargets methods. [4 marks]
* SCREEN CAPTURE(S) showing the required tests. [1 mark]

**Task 19** **Marks:** 14

This question extends the functionality of the Skeleton Program by introducing an auto solver which tests **all** the possible combinations of NumbersAllowed and standard operators in an expression, and offers suggestions to the user for different expressions which could be entered to identify targets.

Create new functionality in the program to offer suggestions for expressions which identify targets. The program should prompt the user if they would like to receive suggestions in each turn. If selected, the program should display a list of expressions, together with what they evaluate to, onto the screen. If, in testing evaluations, multiple expressions identify the same target, only one of those expressions needs to be displayed to the user.

**What you need to do**

**Task 19.1**

Modify the **PlayGame** method to prompt the user as described and display a list of identified targets with their associated expressions.

Create a new method GenerateEvaluations which calculates possible solutions to targets using the values in the NumbersAllowed list, to operate as described.

**Task 19.2**

Test that the changes you have made work:

* Run the Skeleton Program.
* Press enter to start a standard random game.
* Show the program displaying a prompt asking the user if they would like helper suggestions.
* Show the program displaying expressions and associated evaluations which could be used to identify targets in the Targets list.

**Evidence that you need to provide:**

* Your PROGRAM SOURCE CODE showing the amended **PlayGame** method. [6 marks]
* Your PROGRAM SOURCE CODE showing the new GenerateEvaluations method   
  and any additionally created methods required. [7 marks]
* SCREEN CAPTURE(S) showing the required tests. [1 mark]