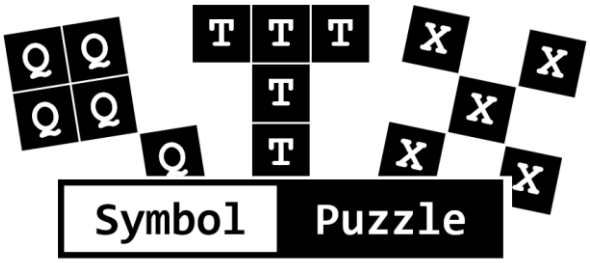


Programming Tasks

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.

AQA has highlighted in the pre-release material that there are errors in the implementation of the puzzle in the Skeleton Program, meaning that it does not work as described under all circumstances. Unless specified, the questions and answers in this document do not correct the errors in the implementation supplied by AQA and, therefore, any anomalous output as a result of these errors will also be present in the solutions supplied as part of this resource.

The objective of this resource is to provide you with a selection of different questions and solutions to those questions. Questions which may have a similar theme may use different techniques to give students a range of options on how to solve problems. Some questions may contain a wider range of development / modification to the code than is realistic in an exam situation to stretch students – in particular Question 18. Equally, some questions are more prescriptive than others in how the task should be completed in order to support a range of learners. The solutions presented in this resource only use techniques which are included in the AQA 7517 specification.

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.

 © ZigZag Education, 2023

Task 1 — Marks: 3

This question refers to the **Puzzle** class.

Introduce a new pattern option into the constructor of the **Puzzle** class for the standard puzzle.

What you need to do

Task 1.1

Add a new pattern option into the puzzle to allow the pattern shown in Figure 1 to be used in a standard puzzle.

(This new pattern option is not expected to work with the default puzzle files supplied by AQA.)

Figure 1

C	C	C
C		
C	C	C

Task 1.2

Test that the changes you have made work:

- Run the Skeleton Program.
- Press `enter` to start a standard puzzle.
- Input the new pattern into an available space on the grid.
- Show the program displaying the new pattern in a standard puzzle and the score increasing by 10 points.

Evidence that you need to provide:

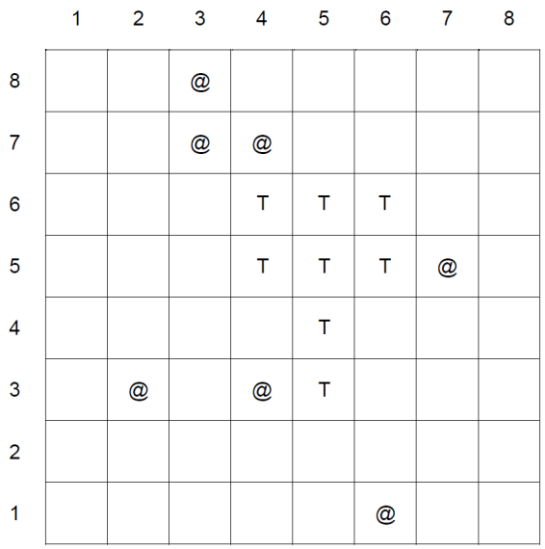
- Your PROGRAM SOURCE CODE showing the introduction of a new pattern option in the constructor of the **Puzzle** class. [2 marks]
- SCREEN CAPTURE(S) showing the required test. [1 mark]

 © ZigZag Education, 2023

Task 2 — Marks: 4

If symbols are placed into the grid in the correct order, it is possible to get patterns which use the same symbol to overlap as shown in the example in Figure 1. In this example, the symbols are entered from the top row down, thereby matching the top T pattern and awarding the user 10 points. When the final T symbol is entered at location 3,5 the lower T pattern is matched, using some of the upper T pattern cells, gaining another 10 points. The legality of this type of move in the pre-release material is not specified and it technically contravenes the rule that when the user has successfully created an allowed pattern, they can no longer place the symbol used in the pattern in any of the other cells in that 3 × 3 section.

Figure 1



This question modifies this logical error in the puzzle so that overlapping patterns created in this way cannot be reused for multiple pattern matches.

The program should still allow the user to place symbols into the grid in the correct order to achieve this overlap effect, but should only award points for a single matched pattern.

What you need to do

Task 2.1

Modify the **CheckforMatchWithPattern** method in the **Puzzle** class to identify if a cell in the pattern helix, which is not blocked, has already been used in a pattern and, if so, return a score of 0.

Task 2.2

Test that the changes you have made work:

- Run the Skeleton Program.
- Press `enter` to create a standard puzzle. *(This may need repeating until a suitable space is shown in the grid to enter the overlapping T patterns shown in Figure 1.)*
- Input a `T` at **Grid** locations `6,4 6,5 6,6 5,4 5,5 5,6 4,5 3,5`. *(These grid locations may need adjusting to suit the space available in your grid.)*
- Show the program displaying the puzzle with overlapping T patterns and a score of 10.

Evidence that you need to provide:

- Your PROGRAM SOURCE CODE for the amended method **CheckforMatchWithPattern** in the **Puzzle** class. [3 marks]
- SCREEN CAPTURE(S) showing the required test. [1 mark]

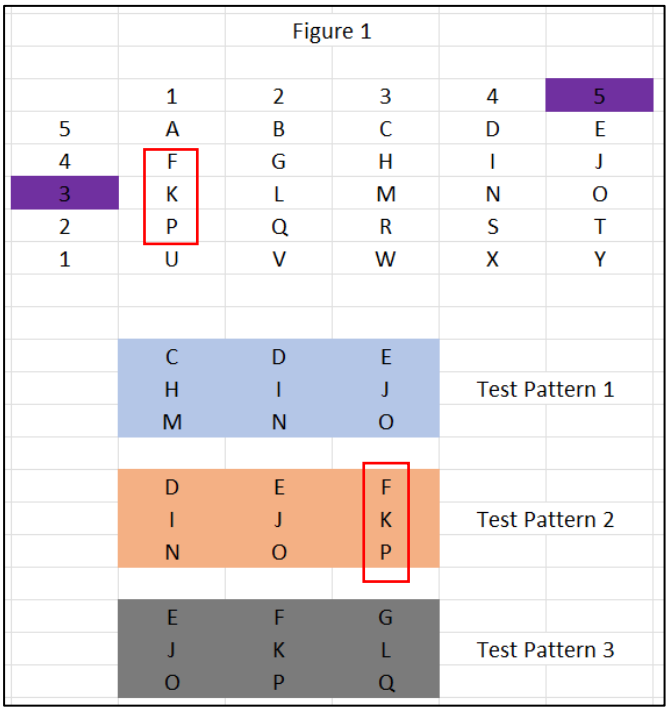
 © ZigZag Education, 2023

Task 3 — Marks: 2

This question modifies the current operation of the program to resolve logic errors. The pre-release material explains that the implementation of the puzzle in the Skeleton Program contains errors, meaning that it does not work in the way described in all circumstances. An error occurs when attempting to match cells where the 3 × 3 section containing those cells is off the right-hand side of the grid. Rather than returning a blank or null cell, the program returns cells which are wrapped around from the left-hand side of the grid on the lines below.

This is shown in Figure 1 where the user has selected grid location 3,5. The program correctly calculates the first pattern to test. The second pattern should contain null values down the right-hand side of the 3 × 3 section because those cells are off the right-hand side of the grid; however, they instead contain the symbols of the wrapped cell one line lower in the grid – highlighted in red.

Figure 1



Given the correct placement of symbols on lower lines, this implementation could generate a logic error resulting in a false positive match of a pattern.

Modify the program to correct this error so that matches can only be found within the correct bounds of the grid.

What you need to do

Task 3.1

Modify the **CheckforMatchWithPattern** method in the **Puzzle** class to calculate the **Grid** locations which are not valid.

Use this information to prevent the method from checking any locations which are not valid in the way described.

Task 3.2

Test that the changes you have made work:

- Run the Skeleton Program.
- Enter `puzzle3`.
- Input a `Q` at **Grid** locations `4,4  4,5  3,4  3,5`.
- Show the program displaying the current score remaining at 10.

Evidence that you need to provide:

- Your PROGRAM SOURCE CODE for the amended method **CheckforMatchWithPattern** in the **Puzzle** class and any other methods you have modified or created when answering this question. [1 mark]
- SCREEN CAPTURE(S) showing the required test. [1 mark]

 © ZigZag Education, 2023

Task 4 **Marks:** 7

This question extends the Skeleton Program to place limits on the number of each type of allowed pattern which the user can place in each puzzle. Currently a user can place as many patterns as they like, subject to having enough symbols and appropriate space in the grid.

Modify the program to select a value at random, up to 3, for the amount of each of the allowed patterns that a user can place into the grid. Each pattern limit may be different. The program should still be limited by the number of symbols. This functionality should only be applicable to standard puzzles. Advise the user when the puzzle is drawn onto the screen of how many of each pattern type they can place.

When a program matches a correctly placed pattern in the grid, the number of patterns that the user can place for that type of pattern should be decremented. The user can still continue to place symbols from that pattern type (subject to having enough symbols left), but the program should no longer match that pattern.

What you need to do

Task 4.1

Modify the constructor in the **Puzzle** class to pass an additional parameter (random between 1 and 3 inclusive) for each pattern when it is instantiated. This is the number of each pattern which can be placed as described.

Task 4.2

Modify the **Pattern** class to use this additional parameter to limit the number of patterns which can be placed into the **Grid**. Store this value in a new property called **PatternCount**. You only need to place restrictions on a standard puzzle. **PatternCount** for the associated pattern should be decremented as each valid pattern is placed.

Task 4.3

Create a new method **OutputPatternCount** in the **Pattern** class which displays an appropriate message onto the screen stating the limit for each pattern type.

Task 4.4

Test that the changes you have made work:

- Run the Skeleton Program.
- Press `enter` to create a standard puzzle.
- Show the program displaying the number of each pattern type available.
- Input a T pattern at a suitable location.
- Show the program displaying the reduction in the number of T pattern types available.
- Repeat the above to use all the T patterns available.
- Show the program giving a suitable error message when the user attempts to place a T symbol.

Evidence that you need to provide:

- Your PROGRAM SOURCE CODE for the amended constructor in the **Pattern** class and the new method **OutputPatternCount** and any other methods you have modified or created when answering this question. [6 marks]
- SCREEN CAPTURE(S) showing the required test. [1 mark]

 © ZigZag Education, 2023

Task 5 **Marks:** 8

This question extends the Skeleton Program to allow the user to remove a symbol from the grid and increase the number of symbols remaining.

A symbol can only be moved if it is not already part of a pattern or is a blocked cell. If the user attempts to move a symbol which is already part of a pattern or is blocked, the program should give them a suitable error message.

The program should display the number of **SymbolsLeft** after the current score has been displayed. The program should ask the user if they would like to remove a symbol. If they select this option, the program should prompt the user for the location of the symbol they want to remove, and if it is a valid location, remove that cell from the grid and increment the number of **SymbolsLeft**. Invalid locations are those which are already part of a matched pattern or are blank or contain a blocked cell. If the target location is not valid, the program should display a suitable error message.

What you need to do

Task 5.1

Modify the **AttemptPuzzle** method in the **Puzzle** class to display the number of **SymbolsLeft** and prompt the user to remove a symbol in the way described.

Task 5.2

Test that the changes you have made work:

- Run the Skeleton Program.
- Enter `puzzle3`.
- Confirm that you want to remove a symbol when prompted to do so.
- Remove the X symbol from **Grid** location `2,3`.
- Show the program displaying the symbol removed from the **Grid** and the **SymbolsLeft** increasing to 11.
- Attempt to remove the Q symbol from **Grid** location `5,1`.
- Show the program displaying a suitable error message.

Evidence that you need to provide:

- Your PROGRAM SOURCE CODE for the amended **AttemptPuzzle** method in the **Puzzle** class and any other methods you have modified or created when answering this question. [7 marks]
- SCREEN CAPTURE(S) showing the required test. [1 mark]

 © ZigZag Education, 2023

Task 6 **Marks:** 9

This question extends the Skeleton Program to allow the user to save a puzzle in its current state.

The program should give the user the option to save the current state of their puzzle to an external file after each turn. On confirmation, the program should ask the user for a filename without the file extension, which the program should append automatically. The program can assume that the filename and location is valid and is a new file. The program should collect together all of the required information for a puzzle in the correct order and save a valid puzzle file.

What you need to do

Task 6.1

Create a new method called **SavePuzzle** in the **Puzzle** class which saves the current puzzle correctly as per the layout of puzzle files supplied by AQA.

Task 6.2

Modify the **AttemptPuzzle** method in the **Puzzle** class to prompt the user after each turn if they would like to save the current puzzle, so that if the user confirms, the new method **SavePuzzle** is called and the puzzle is saved correctly.

Task 6.3

Test that the changes you have made work:

- Run the Skeleton Program.
- Enter `puzzle3`.
- Input an `X` at **Grid** location `1,4`.
- Enter the filename `MySavedPuzzle`.
- Show a screenshot of the exported text file.

Evidence that you need to provide:

- Your PROGRAM SOURCE CODE for the new method **SavePuzzle** in the **Puzzle** class and any other methods you have modified or created when answering this question. [8 marks]
- SCREEN CAPTURE(S) showing the required test. [1 mark]

 © ZigZag Education, 2023

Task 7 **Marks:** 8

This question extends the Skeleton Program by allowing the user to undo previous moves in the puzzle. The user should be able to undo as many previous moves as they have played in the puzzle. Undoing a previous move where points were awarded for a pattern match does not need to remove the points from the score.

Introduce new functionality to store moves made by the user in an appropriate data structure to allow them to be undone. Before each turn, the user should be offered the option to undo their previous move. If no previous moves are available, the program should operate as normal to allow the user to make a move. The program should display to the user how many undo moves they have available. If the user selects to undo, the program should undo their previous move. **Score** changes do not need to be taken into consideration.

What you need to do

Task 7.1

Create a new class **PreviousMove** which inherits **Cell**. A **PreviousMove** should contain appropriate properties together with accessor and mutator methods to store the location and **Cell** from a previous move. All appropriate information about a cell needs to be stored, such as the **Symbol** and the **SymbolsNotAllowed** list.

Task 7.2

Create a new method in the **Puzzle** class called **UndoPreviousMove** which operates in the way described.

Task 7.3

Test that the changes you have made work:

- Run the Skeleton Program.
- Enter `puzzle3`.
- Input a `T` at **Grid** location `4,4`.
- Show the program displaying the updated **Grid** with the new **Symbol** placed.
- Confirm to undo previous move when prompted at next turn.
- Show the program displaying the updated **Grid** with the **Symbol** removed and original replaced.

Evidence that you need to provide:

- Your PROGRAM SOURCE CODE for the new class **PreviousMove**. [2 marks]
- Your PROGRAM SOURCE CODE for the new method **UndoPreviousMove** and the amended method **AttemptPuzzle** and any other methods you have modified or created when answering this question. [5 marks]
- SCREEN CAPTURE(S) showing the required test. [1 mark]

 © ZigZag Education, 2023

Task 8 **Marks:** 9

This question extends the Skeleton Program by allowing the user to move all the blocked cells around to new random locations during the puzzle.

New functionality should be introduced which offers the chance to reshuffle all the blocked cells around after a pattern has been successfully matched on the grid. On selecting this option, the puzzle should move all the blocked cells to new random locations in the grid. A blocked cell cannot be placed back where there was one originally and can only be placed in an empty cell.

What you need to do

Task 8.1

Create a new method in the **Puzzle** class called **ReShuffleBlockedCells** which operates in the way described.

Task 8.2

Modify the **AttemptPuzzle** method in the **Puzzle** class to call the new **ReShuffleBlockedCells** method when a pattern has been successfully matched in the **Grid**.

Task 8.3

Test that the changes you have made work:

- Run the Skeleton Program.
- Enter `puzzle4`.
- Input a `T` at **Grid** locations `4,2  4,3  4,4  3,3  2,3`.
- Show the program displaying the updated **Grid** with the completed pattern placed.
- Select the option to move each **BlockedCell** to a new random location.
- Show the program displaying the updated **Grid** with the random **BlockedCell** locations.

Evidence that you need to provide:

- Your PROGRAM SOURCE CODE for the new method **ReShuffleBlockedCells** and the amended method **AttemptPuzzle.** [8 marks]
- SCREEN CAPTURE(S) showing the required test. [1 mark]

 © ZigZag Education, 2023

Task 9 **Marks:** 8

This question extends the Skeleton Program by introducing a double points cell. The additional functionality should only be available when a standard puzzle is generated. Five double points cells should be added to a grid at random empty locations. A double points cell is shown as a 'D' symbol on the grid.

If a pattern is matched and any of the symbols within that pattern are on a double points cell, the pattern is awarded 20 points rather than 10.

What you need to do

Task 9.1

Create a new class **DoublePointCell** which should inherit from the **Cell** class. A **DoublePointCell** should have the **Symbol** 'D' and should have an overridden method called **IsDouble** which returns true. Include the required virtual method in the **Cell** class to allow this to operate.

Task 9.2

Modify the appropriate constructor for the **Puzzle** class for a standard puzzle to add five double points cells to the **Grid** at empty locations.

Task 9.3

Modify the **CheckforMatchWithPattern** method in the **Puzzle** class so that it additionally matches patterns on a double points cell in the way described.

Task 9.4

Test that the changes you have made work:

- Run the Skeleton Program.
- Press `enter` to create a standard puzzle.
- Input **Symbols** to create a pattern, either 'Q', 'T' or 'X', ensuring that at least one symbol is placed on a **DoublePointCell** cell.
- Show the program displaying the score of 20 when the pattern is matched.

Evidence that you need to provide:

- Your PROGRAM SOURCE CODE for the new class **DoublePointCell**, the amended constructor in the **Puzzle** class and the amended **CheckforMatchWithPattern** method. [7 marks]
- SCREEN CAPTURE(S) showing the required test. [1 mark]

 © ZigZag Education, 2023

Task 10 **Marks:** 7

This question extends the Skeleton Program to include error handling. The pre-release material explains that the Skeleton Program should allow a user to place a different symbol into a pattern, but only in empty cells within the 3 × 3 pattern section. The implementation of the program currently allows a user to place a different symbol into any cell within the 3 × 3 section of a matched pattern, assuming it is not a blocked cell. Additionally, the program gives an unhandled exception if the user selects a grid location outside the bounds of the grid. If the user attempts to enter a symbol at grid location of `GridSize + 1, GridSize + 1`, the program places the symbol at location `GridSize, 1`, which on a standard puzzle grid is location 8,1 and with one of the test puzzle files is location 5,1.

The program should give suitable error messages under the following conditions. The program should not decrement the number of symbols left and instead just give the user a prompt to re-enter a new grid location:

- The user attempts to place the same symbol into **any** cell within the 3 × 3 section of a matched pattern.
- The user attempts to place a different symbol into any **non-empty** cell within the 3 × 3 section of a matched pattern.
- The user attempts to enter a grid location which is outside the bounds of the grid or is in an invalid format.

What you need to do

Task 10.1

Modify the **Cell** class to include appropriate accessor and mutator methods to identify it as being part of a pattern.

Task 10.2

Modify the **AttemptPuzzle** method in the **Puzzle** class to advise the user how many symbols they have left and introduce suitable error handling so that the user cannot place a symbol in an invalid location as described.

Task 10.3

Test that the changes you have made work:

- Run the Skeleton Program.
- Enter `puzzle3`.
- Attempt to input a `Q` at **Grid** location `4,3`.
- Show the program giving an error message, prompting the user to try again and the number of symbols not changing.
- Attempt to input an `X` at **Grid** location `5,1`.
- Show the program giving an error message, prompting the user to try again and the number of symbols not changing.
- Attempt to input an `X` at **Grid** location `9,9`.
- Show the program giving an error message, prompting the user to try again until they enter a valid value.

Evidence that you need to provide:

- Your PROGRAM SOURCE CODE for the amended **Cell** class and the amended method **AttemptPuzzle** in the **Puzzle** class and any other methods you have modified or created when answering this question. [6 marks]
- SCREEN CAPTURE(S) showing the required test. [1 mark]

 © ZigZag Education, 2023

Task 11 **Marks:** 6

This question refers to the **AttemptPuzzle** method.

Introduce new functionality to award the user a lever. A lever can remove a blocked cell completely from the grid, allowing the user to then place a normal symbol into that location in a later turn. A lever is awarded to the user after they have successfully placed their first valid pattern. The user should only be able to use the lever once in a puzzle. On selecting to use the lever, the user should be prompted to enter the grid location of the blocked cell which they want to remove. A valid location is one which contains a blocked cell. Assuming it is a valid location, replace the blocked cell with a normal cell, allowing the user to place a symbol at that location in a later turn.

What you need to do

Task 11.1

Modify the **CheckforMatchWithPattern** method to award the user with a lever on placing a valid pattern.

Task 11.2

Modify the **AttemptPuzzle** method so that it operates in the way described.

Task 11.3

Test that the changes you have made work:

- Run the Skeleton Program.
- Enter `puzzle3`.
- Input an `X` at **Grid** location `1,4`.
- Select to use the lever when prompted.
- Enter **Grid** location `5,3`.
- Show the program displaying the **Grid** with the **BlockedCell** removed.
- Input an `X` at **Grid** location `5,3`.
- Show the program displaying the symbol `X` at the new location.

Evidence that you need to provide:

- Your PROGRAM SOURCE CODE for the amended **AttemptPuzzle** and **CheckforMatchWithPattern** methods and any other methods you have modified or created when answering this question. [5 marks]
- SCREEN CAPTURE(S) showing the required test. [1 mark]

 © ZigZag Education, 2023

Task 12 **Marks:** 8

This question refers to the **AttemptPuzzle** method in the **Puzzle** class and the modification of the **BlockedCell** class to create a swamp of blocked cells. A swamp is an individual **BlockedCell** but with the symbol '!'. Multiple swamps can be placed into the grid to make placement of patterns more difficult.

Introduce new functionality to fill some empty space on the grid with swamps. Swamps are placed at random locations in the grid. The swamp can only be placed into a cell which is currently empty and has the symbol '!' to identify the cells as being a swamp rather than a normal blocked cell. Swamps fill a random number of empty cells (between 1 and 4 inclusive). There should be a 25% chance of swamps being triggered in each turn. The user must be given between 2 and 4 (chosen at random) turns warning of a swamp event. Swamps can only be triggered once in a puzzle.

What you need to do

Task 12.1

Modify the **BlockedCell** class to allow the symbol to be updated to '!' by overriding and adding code to the **Update** method.

Task 12.2

Create a new method in the **Puzzle** class called **SwampThisCell** which inserts a new **BlockedCell** into the **Grid** in the way described.

Task 12.3

Modify the **AttemptPuzzle** method to operate in the way described.

Task 12.4

Test that the changes you have made work:

- Run the Skeleton Program.
- Enter `puzzle4`.
- Place symbols into the **Grid** until a swamp is triggered.
- Show the program advising the user that a swamp has been triggered, how many turns are left until the swamp happens and how many cells will be filled as swamps.
- Continue to place symbols into the **Grid** until the swamp happens.
- Show the program displaying the correct number of randomly placed swamp cells.

Evidence that you need to provide:

- Your PROGRAM SOURCE CODE for the new method **SwampThisCell** and the amended method **AttemptPuzzle**. [7 marks]
- SCREEN CAPTURE(S) showing the required test. [1 mark]

 © ZigZag Education, 2023

Task 13 **Marks:** 12

This question extends the Skeleton Program by checking if there are enough symbols left to be able to place another pattern into the grid. If there are not enough symbols left to be able to place another pattern into the grid, the number of **SymbolsLeft** should be deducted from the player **Score** and the puzzle finishes. This question does not correct the error in the Skeleton Puzzle which allows different symbols to be placed on top of currently completed patterns or cells with the same symbol to be included in multiple pattern matches, but it does need to take into account the normal rules of the puzzle, e.g. the system cannot attempt to place a symbol into a blocked cell.

After each turn the program should calculate how many patterns there are in the grid and test to see which ones match allowed patterns. For patterns which are only partially matched, the program should test to see how many symbols are currently in the right place and whether there are enough symbols left over to complete the pattern.

If there are enough symbols left to complete a pattern in the grid, the puzzle should continue with the user's next turn. If there are not enough symbols, the number of **SymbolsLeft** should be deducted from the **Score** and the puzzle should finish. The program should tell the user what the deduction is and their final score.

What you need to do

Task 13.1

Create a new method in the **Puzzle** class called **GetSpaceLeftOnGrid** that calculates if there are enough symbols left to add a new pattern to the grid in the way described.

Task 13.2

Modify the **AttemptPuzzle** method in the **Puzzle** class which uses the new method **GetSpaceLeftOnGrid** to check if the user can enter another pattern as described.

Task 13.3

Test that the changes you have made work:

- Run the Skeleton Program.
- Enter `puzzle2`.
- Input an `X` at **Grid** location `1,4`.
- Show the program advising the user that there are not enough symbols to complete any more patterns in this grid and 2 points will be deducted from their final score.
- Show the program displaying a finished puzzle with a final score of 18.

Evidence that you need to provide:

- Your PROGRAM SOURCE CODE for the new method **GetSpaceLeftOnGrid** and the amended method **AttemptPuzzle** in the **Puzzle** class and any other methods you have modified or created when answering this question. [11 marks]
- SCREEN CAPTURE(S) showing the required test. [1 mark]

 © ZigZag Education, 2023

Task 14 — Marks: 10

This question modifies the Skeleton Program so that it identifies errors in the external puzzle files. Errors in these files generate logic errors when trying to place symbols into the grid.

The files contain two types of error:

- Error 1: A cell contains a **SymbolsNotAllowed** list but does not appear to be part of a pattern.
- Error 2: A cell does not contain a **SymbolsNotAllowed** list when it appears it should be part of a pattern.

Introduce new functionality which, after loading a puzzle file, checks the data for the errors shown above. The program should advise the user if the file demonstrates an error. It does not need to show where.

Puzzle Files	Error	Result
Puzzle4.txt	3 Q T X 3 Q,QQ**Q**QQ X,X*X*X*X*X T,TTT**T**T 5 ,X,Q , @, , , **Error 1**	The 10th line in the puzzle file shows a **SymbolsNotAllowed** list for the empty cell at location 5,1 in the grid. If a cell is part of a pattern, this scenario is possible; however, the puzzle file contains no patterns. This generates the logic error that the user cannot place a Q or an X symbol at this location.
Puzzle1.txt Puzzle2.txt Puzzle3.txt	5 Q,Q Q,Q @,Q , , Q,Q Q,Q ,Q , , X,Q Q,Q **Error 2**	The 20th line in the puzzle files has an empty **SymbolsNotAllowed** list for the empty cell at location 3,1 in the grid. This cell is part of the matched Q pattern in the 3 × 3 section from location 5,1 to 3,3. This generates the logic error that the user can place a Q symbol at this location, which they should not be able to do.

 © ZigZag Education, 2023

What you need to do

Task 14.1

Create two new methods called **TestForError1** and **TestForError2** that iterate through the **Grid** identifying cells which contain the import errors highlighted on the previous page.

Error 1 is demonstrated if cells:

- contain a populated **SymbolsNotAllowedList** and are not a blocked cell
- the game **Score** is zero

Error 2 is demonstrated if:

- the number of cells containing a **SymbolsNotAllowedList** is less than the number that should be given the **Score**

Task 14.2

Modify the **LoadPuzzle** method in the **Puzzle** class to call the **TestForError1** and **TestForError2** methods once all the puzzle data has been loaded, and display the results in the way described.

Task 14.3

Create a new method **ContainsSymbolsNotAllowedList** in the **Cell** class which returns true if the cell is a blocked cell or if the **SymbolsNotAllowed** list contains at least one element but does not contain an empty string.

Task 14.4

Test that the changes you have made work:

- Run the Skeleton Program.
- Enter `puzzle4`.
- Show the program indicating that the puzzle demonstrates error 1.
- Run the Skeleton Program again.
- Enter `puzzle1`.
- Show the program indicating that the puzzle demonstrates error 2.

Evidence that you need to provide:

- Your PROGRAM SOURCE CODE for the new methods **TestForError1** and **TestForError2** in the **Puzzle** class and the amended **LoadPuzzle** method together with the new **ContainsSymbolsNotAllowed** method in the **Cell** class. [8 marks]
- SCREEN CAPTURE(S) showing the required test. [2 marks]

 © ZigZag Education, 2023

Task 15 **Marks:** 9

This question extends the Skeleton Program to allow rotated patterns to be matched. In the pre-release material, the puzzle will only match a pattern which has been entered in a normal configuration; for example, the Q pattern shown in Figure 1.

Figure 1

Q	Q	
Q	Q	
		Q

Introduce new functionality into the program to allow the puzzle to additionally match patterns rotated at 90, 180 and 270 degrees; for example, the Q patterns shown in figures 2, 3 and 4. If the user correctly places a rotated pattern, they should be awarded 15 points rather than just 10. *Due to the wrap-around error in the pre-release code, some rotations may be identified as false positives. This question does not need to correct that error.*

Figure 2

	Q	Q
	Q	Q
Q		

Figure 3

Q		
	Q	Q
	Q	Q

Figure 4

		Q
Q	Q	
Q	Q	

Character positions for a normal Q pattern:

1	2	3	4	5	6	7	8	9
Q	Q	*	*	Q	*	*	Q	Q

A 90 degree rotation uses the normal pattern sequence in the following order:
chars 7 and 8 followed by the first 6 chars followed by the final char, e.g.: ****QQQ**Q*Q***

A 180 degree rotation uses the normal pattern sequence in the following order:
chars 5 to 8 followed by the first 4 chars followed by the final char, e.g.: ***Q**QQQ**Q***

A 270 degree rotation uses the normal pattern sequence in the following order:
chars 3 to 9 followed by the first 2 chars, e.g.: *****Q**QQQQ***

What you need to do

Task 15.1

Create a new method called **MatchesPatternRotated** in the **Puzzle** class that identifies a rotated pattern match as described.

Task 15.2

Modify the **CheckforMatchWithPattern** method in the **Puzzle** class to additionally test for rotated patterns by calling the **MatchesPatternRotated** method for each pattern combination. If a rotated pattern is found, award the user 15 points.

 © ZigZag Education, 2023

Task 15.3

Test that the changes you have made work:

- Run the Skeleton Program.
- Enter `puzzle4`.
- Input a T pattern rotated through 90 degrees at a suitable location.
- Show the program awarding a score of 15 when the rotated pattern is matched.

Evidence that you need to provide:

- Your PROGRAM SOURCE CODE for the new method **MatchesPatternRotated** and the amended method **CheckforMatchWithPattern** and any other methods you have modified or created when answering this question. [8 marks]
- SCREEN CAPTURE(S) showing the required test. [1 mark]

 © ZigZag Education, 2023

Task 16 **Marks:** 9

This question modifies the Skeleton Program to adjust how scoring is handled when different symbols are placed into a non-empty cell which has already been matched and scored as part of a pattern. The pre-release material indicates that this should not be possible; however, this is not how it has been implemented. In the current implementation, if the user places a different symbol into a non-empty cell which is part of a pattern, the cell symbol is replaced and the user does not lose the points gained from that pattern, even though the pattern is no longer correct.

Introduce new functionality which removes points awarded to a player if they place a different symbol into a non-empty cell which is part of a pattern, thereby making the pattern no longer valid.

What you need to do

Task 16.1

Modify the **Cell** class to include appropriate mutator and accessor methods to allow a **Cell** to be set, unset and tested as being part of a pattern when a new pattern is matched. Include an appropriate mutator method which removes a symbol from the **SymbolsNotAllowedList**.

Task 16.2

Modify the **CheckforMatchWithPattern** method to pass in an additional parameter to operate as a flag to indicate if the method should operate as normal or to remove symbols from the **SymbolsNotAllowedList** and decrease the **Score**. Modify the **CheckforMatchWithPattern** method to use this flag and operate as described.

Task 16.3

Create a new method in the **Puzzle** class called **RemoveSymbol** which should be called from the **AttemptPuzzle** method and change the symbol and update **Score** as described.

Task 16.4

Test that the changes you have made work:

- Run the Skeleton Program.
- Press `enter` to create a standard puzzle.
- Input a valid `T` pattern into the **Grid** at an available location.
- Show the program displaying the updated **Score** of 10.
- Input a `Q` symbol into the **Grid** into a cell containing one of the T symbols.
- Show the program displaying the updated **Score** of 0.

Evidence that you need to provide:

- Your PROGRAM SOURCE CODE for the modifications in the **Cell** class. [3 marks]
- Your PROGRAM SOURCE CODE for the new method **RemoveSymbol** and the amended method **AttemptPuzzle** and any other methods you have modified or created when answering this question. [5 marks]
- SCREEN CAPTURE(S) showing the required test. [1 mark]

 © ZigZag Education, 2023

Task 17 **Marks:** 11

This question extends the Skeleton Program by correcting the error in the implementation which allows symbols of different types to be placed into non-empty cells which have already been matched and scored as part of a pattern. This error, however, should be corrected to allow the concept of a wild card cell. When the user attempts to place a different symbol onto a non-empty cell which has already been matched and scored as part of a pattern, the program should instead place a wild card cell. A wild card has the symbol 'W' and can represent any symbol in the **AllowedSymbolsList**. When a pattern is matched which includes a wild card it should gain 15 points.

A wild card cannot be placed onto a blocked cell. A user can have only two wild cards in each puzzle. The program should tell the user how many wild cards they have left at each turn. When a user has placed both wild cards, give a suitable error message if they attempt to place another symbol into a non-empty cell which has already been matched and scored as part of a pattern.

What you need to do

Task 17.1

Create a new **WildCardCell** class which inherits **Cell**. It should have the **Symbol** property of 'W'. It should have an appropriate accessor method to identify it as a wild card.

Task 17.2

Modify the **AttemptPuzzle** method in the **Puzzle** class and any other methods required. If the user attempts to place a different symbol into a non-empty cell which has already been matched and scored as part of a pattern, advise the user that the cell is part of a pattern and place a **WildCardCell** at that location instead which should operate in the way described.

Task 17.3

Test that the changes you have made work:

- Run the Skeleton Program.
- Enter `puzzle3`.
- Input an `X` at **Grid** location `1,4`.
- Input a `T` at **Grid** locations `4,3  4,4  4,5  3,4  2,4`.
- Show the program displaying the puzzle with a T pattern overlapping the X pattern with a wild card at **Grid** location 3,4 and a score of 35.

Evidence that you need to provide:

- Your PROGRAM SOURCE CODE for the new class **WildCardCell** and any other classes modified when answering this question. [3 marks]
- Your PROGRAM SOURCE CODE for the amended method **AttemptPuzzle** in the **Puzzle** class and any other methods you have modified or created when answering this question. [7 marks]
- SCREEN CAPTURE(S) showing the required test. [1 mark]

 © ZigZag Education, 2023

Task 18 **Marks:** 16

Warning! This question requires the student to generate a large amount of code which is beyond what is realistic in an exam scenario.

This question extends the program by showing the user a possible solution for achieving the highest score possible from placing legal patterns into the grid. This should operate as an auto complete function for an empty grid. This question is **not** designed to auto complete a partially completed grid.

A legally placed pattern cannot overlap with another pattern of the same symbol type at all. Patterns of different symbol types can overlap, but not if the overlap is on non-empty cells which have already been matched and scored as part of a pattern.

The program should calculate possible solutions for achieving the highest score possible. The program can assume an unlimited number of symbols. Multiple solutions may achieve the highest score possible. The program only needs to show one of them to the user together with the score it achieves.

After loading an empty puzzle file (puzzle file 4), the program should ask the user if they would like to auto complete the puzzle. The program should calculate possible solutions to the puzzle. It should then display one of the highest scoring possible solutions to the user together with the score it achieves. The program should then finish.

What you need to do

Task 18.1

Create a new **TempPuzzle** class which should contain all the required properties to store a completed puzzle together with the associated accessor and mutator methods.

Task 18.2

Create a new method in the **Puzzle** class called **CheckAllCombinations** which tests all the possible combinations of 'Q', 'X' and 'T' patterns in the grid to work in the way described. It should return a suitable data structure which contains all the possible combinations together with the score. Display one of the highest scoring possible solutions to the user together with the score it achieves.

Task 18.3

Test that the changes you have made work:

- Run the Skeleton Program.
- Enter `puzzle4`.
- Confirm that you would like the program to auto complete the puzzle when prompted.
- Show the program displaying the completed puzzle with an overlapping 'Q' and 'X' pattern and the score of 20.

Evidence that you need to provide:

- Your PROGRAM SOURCE CODE for the new class **TempPuzzle**. [3 marks]
- Your PROGRAM SOURCE CODE for the amended method **CheckAllCombinations** in the **Puzzle** class and any other methods you have modified or created when answering this question. [12 marks]
- SCREEN CAPTURE(S) showing the required test. [1 mark]

 © ZigZag Education, 2023