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]

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]

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]

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]

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]

