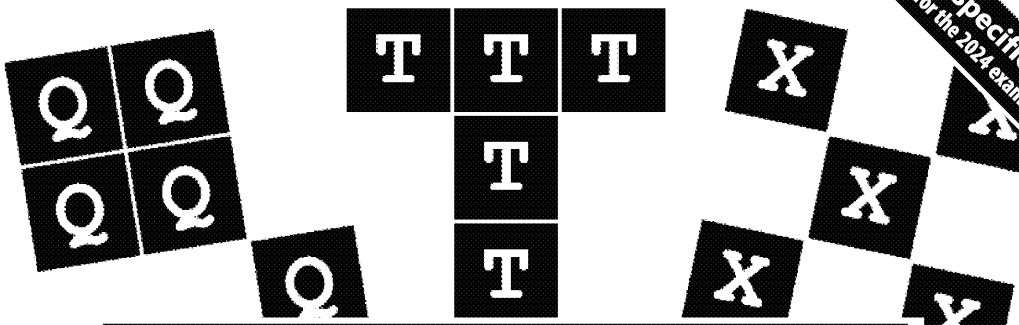
Zig Zag Education

Computer Science

A Level | AQA | 7517

2015 specification for the 2024 exam



Symbol Puzzle

PAPER 1 EXAM RESOURCE PACK 2024

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 12193

zigzageducation.co.uk

Contents

Printouts of electronic resources (for reference)

** Note there are also electronic copies of the UML Diagrams ('Complete' & 'Activity' versions) which can be printed in A3, making them much more usable (especially when used as activities)*

Teacher's Introduction

Symbol Puzzle is a single-player puzzle game where the user needs to place symbols into a grid, building up patterns. Each time the user places a new symbol into the grid, the application compares cells in the grid, looking for pattern matches. The objective of the program is to get the highest score possible by adding valid patterns into the grid.

The application can either be played using an 8 × 8 standard puzzle grid, or load in a 5 × 5 external file with a partially complete puzzle.

The user can place Q, T or X symbols into empty cells in the grid, attempting to make larger patterns of these letters. Each valid matched pattern awards the user 10 points. The application has a fixed number of symbols available – either 64 for the standard puzzle or varying amounts, dependent on which external puzzle file the user loads. A user can place each symbol as a Q, a T or an X into the grid, subject to there being space and subject to the placement rules.

Patterns are matched in a 3 × 3 section of the grid – a pattern is nine cells in total in a specific order. Once a valid match for a specific symbol has been identified, the cells in that 3 × 3 section are modified so that the same symbol cannot be placed into any of the cells in that 3 × 3 section of the grid in a future move. This prevents overlapping patterns of the same symbol type being placed into the grid. Symbols of a different type can be placed into those cells, however, allowing for patterns of a different symbol type to overlap. The grid can also contain blocked cells – denoted by an '@' symbol. The user cannot place a symbol into these cells.

When the user has exhausted all their symbols, the puzzle is complete.

This resource aims to help you get to grips with and prepare for the A Level Paper 1 examination for summer 2024, which is partly based on the ***Symbol Puzzle*** 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:

`SymbolPuzzle`	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.

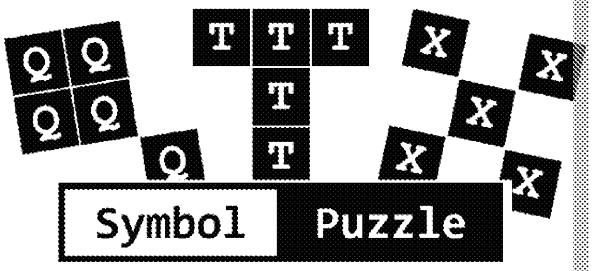
Installation: Extract the files from the downloaded ZIP file and move the entire `SymbolPuzzle` folder onto a network location that is accessible for students, and provide them with a shortcut to the index.html file. All content can be accessed from this page.

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-Diagam-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. In addition to the code breakdown, there is also a breakdown of the puzzle files, errors in the code, and a set of 'advanced techniques' for exploring the code further.
- **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. There are 10 in total.
- **Video**: a quick overview of the ***Symbol Puzzle*** 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 18 modification tasks to test your programming skills – as well as an additional 12 modification ideas that you may also want to try as extension tasks.



Skeleton Code Breakdow

Static Methods

Identifier / Data	Description
Main	
Parameters:	n/a
Return value:	n/a

This is the main entrance point fo
determine whether the application
randomly created puzzle or load a

It initialises a string variable called
and an integer variable of Score.

The method then enters into the n
in that loop by the value of the Ag
using the following steps:

- Prompt the user if they would
puzzle or load an external file
- Instantiate a new Puzzle obje
- If the user has entered a filen
constructor in the Puzzle clas
parameter. This will create a
text file.
- If the user does not enter a fil
the length of the filename bei
constructor in the Puzzle clas
38. The first parameter is the
puzzle, creating a grid which
The second parameter is calc
GridSize, multiplying the resu
down to an integer value.
- The method then calls the At
MyPuzzle which starts the pu
wh puzzle is complete is
Sc variable which is displa
The method then prompts the
another puzzle, setting the Ag
either stop the loop and, there
allow it to repeat again.

Class: Puzzle

Identifier / Data		Description
<<constructor>>		
Parameters	Filename : String	Initialises the following private attri • Grid as List of Cells • AllowedPatterns as List of P • AllowedSymbols as List of S These attributes are subsequently LoadPuzzle() method. The method calls the LoadPuzzle filename parameter. **This constructor is called when loaded.**
Return values	n/a	
<<constru		
Parameters	Size : Int StartSymbols : Int	Initialises the following private attri • Score to 0 • SymbolsLeft from paramete • GridSize from parameter Siz • Grid as List of Cells The method performs a count-con square of the GridSize. Inside the adds all the cells for a standard pu random number between 1 and 1 of being a normal cell and a 10% c The method then initialises Allow and AllowedSymbols as a list of default pattern objects for the Q, X to the AllowedPatterns list togeth symbol 'Q', 'X' or 'T' to the Allowe **This constructor is called when generated.**
Return values	n/a	

AttemptPuzzle (public) <<virtual>>		
Parameters	n/a	This method is the main loop for at It is held in the loop using the local The method firstly displays the cu DisplayPuzzle() method. It then c The method then asks the user to Column of where they would like puzzle. The method uses try…cat user has entered integer values, b valid within the bounds of the grid error mess[illegible] to the user for er The m[illegible]d then calls the GetSy a symbol to place into the grid fro number of symbols available left t The method then creates a local c given by the user and tests to see can be used in that cell by calling method passing the symbol enter If the symbol can be used in that c changed using the ChangeSymb The method then checks if this up by calling the CheckForMatchWi Row and Column entered by the The result of this check is assigne AmountToAddToScore. If this is the user Score. The method then checks if all the used, and if so, exits the main pro If the main program has exited, the calling the DisplayPuzzle() method
Return values	Score : Int	
CheckForMatchWithPattern (public) <<virtual>>		
Parameters	Row : Int Column : Int	Uses a nested loop to iterate throu together a string variable called Pa from nine cells in a 3 × 3 section of nested loop is used to concatenate of pattern that could include the cel × 3 section of the grid. This is done prevent the code from crashing if a bounds of the grid is accessed. Once the PatternString has been a foreach loop to check it for a ma by calling the MatchesPattern() m PatternString and symbol being c If a match is found, the method ca AddToNotAllowedSymbols() on matching 3 × 3 section of the grid checked as a parameter. This pre being placed into that cell in a futu If a match has been found, the me otherwise it returns the Score of 0
Return values	Int	

CreateHorizontalLine (private)		
Parameters	n/a	Uses iteration to concatenate a ho which is the correct width for the g puzzle.
Return values	Symbol : String	
DisplayPuzzle (public) <<virtual>>		
Parameters	n/a	Used to print out the grid onto the using the following steps: • Print a blank line. • If the GridSize is less than 10 screen, then iterate through to follows by the count heading • Print a blank line. • Print a horizontal line by callin method. • The method then iterates thro printing out the row number fo symbol and the symbol in eac uses the MOD function to cal before printing a final '|' symb underneath. • This process is repeated until printed to the screen.
Return values	n/a	
GetCell (private)		
Parameters	Row : Int Column : Int	Uses the Row and Column param Cell element in the one-dimension returned.
Return values	Cell	
GetSymbolFromUser (private)		
Parameters	n/a	Used for getting a symbol from the The method uses a loop to repeat symbol they want to use until they part of the AllowedSymbols list. When they enter a valid symbol, it
Return values	Symbol : String	

LoadPuzzle (private)		
Parameters	Filename : String	The method loads an external tex parameter. See 'Puzzle File Breakdown' for d an external puzzle file. The method sequences through th the Filename parameter. It perfor Assigns NoOfSymbols from the f this value to iterate through the ne from th e into the AllowedS signs NoOfPatterns from the n this value to iterate through the fo pattern. A pattern line contains a c element is the symbol for a patter the pattern itself. These are used t object which is then added to the Assigns GridSize from the next s square of this value to iterate thro in each cell. A cell line contains a first element is the symbol for the elements is a sublist which contai SymbolsNotAllowed list for that '@' symbol the application instanti BlockedCell into the Grid. Other a Cell by using the cell symbol an allowed elements. Once all the cells have been appe assigns the next line in the file to t line to the SymbolsLeft attribute. The method uses a try…catch stru error occurs, an error message is g does not give the user the opportu
Return values	n/a	

Class: Pattern

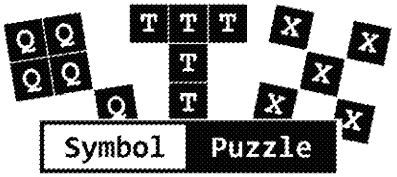
Identifier / Data		Description
<<constructor>>		
Parameters	SymbolToUse : String PatternString : String	Initialises the following private at • Symbol from parameter Sy • PatternSequence from par
Return values	n/a	
GetPatternSequence (public) <<virtual>>		
Parameters	n/a	Returns the value of the private
Return values	PatternSequence : String	
MatchesPattern (public) <<virtual>>		
Parameters	PatternString : String SymbolToUsed : String	This is used to confirm that a pa concatenation through a 3 × 3 s PatternSequence attribute in a
Return va	Boolean	If the passed parameter Symbo Symbol for the pattern, the meth If it does match, the method itera attribute PatternSequence com at the same position in the para PatternSequence contains sym for the pattern, and the '*' charac to match in the pattern. The itera between the PatternSequence parameter which match the sym differences, the pattern match re this iteration uses a try…catch t comparisons of out-of-range err the method displays to the user If the iteration completes, all the matched and, therefore, the met

Class: Cell

Identifier / Data		Description
<<constructor>>		
Parameters	n/a	Initialises the protected attribute
Return values	n/a	Initialises the private attribute Sy empty string list.
AddToNotAllowedSymbols (public) <<virtual>>		
Parameters	SymbolToAdd : String	Appends the parameter Symbol attribute SymbolsNotAllowed.
Return values	n/a	
ChangeSymbolInCell (public)		
Parameters	NewSymbol : String	Assigns the NewSymbol param Symbol.
Return values	n/a	
CheckSymbolIsAllowed (public) <<virtual>>		
Parameters	SymbolToCheck : String	Iterates through the private list a If the parameter SymbolToChe method returns false, otherwise
Return values	Boolean	
GetSymbol (public) <<virtual>>		
Parameters	n/a	The method uses the IsEmpty() empty. If it is empty, the method returns the value of the protecte
Return values	Symbol : String	
IsEmpty (public)		
Parameters	n/a	If the private attribute SymbolsN method returns true, otherwise i
Return values	Boolean	
UpdateCell (public) <<virtual>>		
Parameters	n/a	This method is not used in the p method has been included into t option for a question which crea cell class and overrides its base
Return values	n/a	

Class: BlockedCell (inherits from Cell)

Identifier / Data		Description
<<constructor>>		
Parameters	n/a	Initialises the parent attribute S
Return values	n/a	
CheckSymbolAllowed (public) <<override>>		
Parameters	SymbolToCheck : String	Overrides the CheckSymbolA class. While technically this co to allow a blocked class object SymbolsNotAllowed list attrib to simply return false regardles SymbolToCheck.
Return values	Boolean	



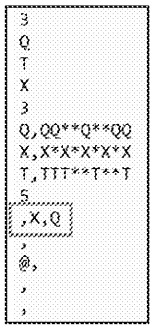
Puzzle File Breakdown of Puzzle1

Line Number	Data	Description
1	3	This is the number of symbols in the game. It is used to iterate through the next three lines.
2	Q	FIRST SYMBOL. This is added to the AllowedSymbols list.
3	T	SECOND SYMBOL. This is added to the AllowedSymbols list.
4	X	THIRD SYMBOL. This is added to the AllowedSymbols list.
5	3	This is the number of patterns in the game. It is used to iterate through the next three lines.
6	Q,QQ**Q**QQ	FIRST PATTERN. This line is imported as a single string then split to lists using the comma as a delimiter. The first element is the Symbol for the pattern, and the second element is the PatternSequence. The PatternSequence represents the pattern as a helix of cells in a 3 × 3 section of the Grid.
7	X,X*X*X*X*X	SECOND PATTERN
8	T,TTT**T**T	THIRD PATTERN
9	5	This is the GridSize. A single value is used to denote the width and the height, i.e. the grid is square. This is used to calculate how many lines in the text files the code then needs to iterate through by multiplying it by itself, in this case 25.
10	Q,Q	The next 25 lines *(because this game has a 5 × 5 grid)* are the symbols in the grid. Each line has two elements. The first element is used to set the symbol in the cell (therefore, one symbol per cell).
11	Q,Q	The second element is what symbol can't be in the cell. This is used to stop two patterns of the same type overlapping. When this is imported,

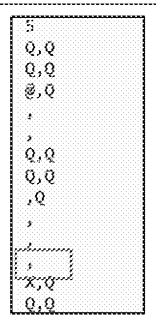
Line Number	Data	Description
17	,Q	This is a blank cell. In this example, however, the cell is within the 3 × 3 section of a matched pattern in the puzzle because it contains a SymbolsNotAllowed list.
18	,	
19	,	
20	,	See the 'Error' description before this row.
21	X,Q	The cell has an X symbol in it; however, the cell is within the 3 × 3 section of a matched pattern in the puzzle because it contains a SymbolsNotAllowed list. This is because the Q pattern was created before the X pattern; therefore, Q was used as the 'SymbolsNotAllowed' symbol.
22	Q,Q	
23	X,	
24	,	
25	,	
26	,	
27	X,	This is showing a cell in the grid during a game. It has an X in it which the user has put into it, but the pattern hasn't yet been complete; therefore, there are currently no symbols banned from this cell.
28	,	
29	,	
30	,	
31	X	
32	,	
33	,	

‘Errors’ in the puzzle files

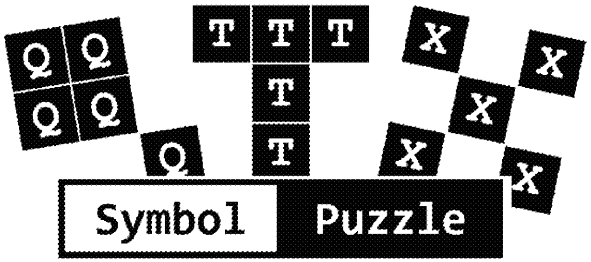
There are two logical errors that we have found in the puzzle files. These ‘errors’ have been checked with AQA, who have confirmed that the files operate as they expect them to or as required for the purpose of the exam. While they don’t impact the main functionality of the application, the ‘errors’ generated unexpected behaviour at runtime.



This ‘error’ is demonstrated in puzzle 4. Line 10 of the file shows an empty cell but a populated ‘SymbolsNotAllowed’ list with an X and a Q. The ‘SymbolsNotAllowed’ list is populated when a user matches a pattern. Under normal operation of the application, it should be logically impossible for a single cell to have populated ‘SymbolsNotAllowed’ list. The impact of this on the application is that when using puzzle 4, the user cannot place an X or a Q symbol into the cell at location 5,1.



This ‘error’ is demonstrated in puzzles 1, 2 and 3. Line 20 of the file shows an empty cell but also an empty ‘SymbolsNotAllowed’ list. The puzzle files demonstrate that a Q pattern has been matched with the top-left cell at location 5,1. This should put a Q into the ‘SymbolsNotAllowed’ list at locations 5,1 to 3,3. Line 20 represents location 3,1 in the grid. Under normal operation of the application, it should be logically impossible for this cell to have a blank ‘SymbolsNotAllowed’ list after a pattern match. The impact of this on the application is that a user can place a Q symbol into location 3,1 when the application should not let them.



Advanced Techniques

These techniques are helper pieces of code to extend the Skeleton Program and functionality.

Note that the techniques demonstrated in this document are **BEYOND** the AQA 7517 specification. They **DO NOT** represent a mark scheme or an expected way of solving challenges presented for the pre-release material. The objective of this document is to extend students' knowledge and explore alternative techniques they could use as they experiment with the code. **Use of these techniques and ideas should be at the teacher's discretion**. The code shown below still demonstrates the 'wrap around' false positive matches identified by the standard skeleton code.

Extension Technique 1 – Regular expressions

The Skeleton Program provided by AQA does not include the regular expression (regex) library and, therefore, it is unlikely that AQA would include a Section D question which uses regex. *(A regex question in Section C is perfectly possible.)* Although the library has not been included, students can import the library themselves and use it if they are confident with regex and feel that its use would aid their solutions.

To use this code you will need to import the regex library: `using System.Text.RegularExpressions;`

An obvious place that could use regex is in the *MatchesPattern* method in the Pattern class, which could be rewritten as:

This expression can be extended further by defining duplicates of symbols:

```
            private bool CheckForMatchWithPatternRegexBeyondStandard(string PatternStringToCheck)
            {
                //These expressions use techniques which are beyond the AQA specification
                //The hyphen must be last otherwise it is interpreted as a range
                Regex TCheck = new Regex("T{3}[Q|X|T|@|-]{2}T[Q|X|T|@|-]{2}T");
                Regex XCheck = new Regex("(X[Q|X|T|@|-]){4}X");
                Regex QCheck = new Regex("Q{2}[Q|X|T|@|-]{2}Q[Q|X|T|@|-]{2}QQ");
                return TCheck.IsMatch(PatternStringToCheck) || XCheck.IsMatch(PatternStringToCheck) ||
QCheck.IsMatch(PatternStringToCheck);
            }
```

Extension Technique 2 – Regular expressions on the grid

The methods shown in technique 1 improve the time complexity of the *MatchesPattern* method from O(n) to O(1). The actual data structure for the pre-release material uses a one-dimensional list; therefore, regex combined with LINQ could be used on this to make the same improvements by removing the need to generate the helix pattern string. The expression, however, needs further modification to take into account that the grid can be different sizes and, therefore, that the space between symbols within a pattern can change. A limitation of this code, however, is that it cannot detect overlapping patterns or some patterns right next to each other because the regex is not identifying overlapping matches.

```
            private bool CheckForMatchWithPatternRegexUsingGrid(string SymbolToCheck)
            {
                string StringRepresentationOfWholeGrid = string.Join("", Grid.Select(cell => cell.GetSymbol()));
                Regex Check = new Regex("");
                //The hyphen must be last otherwise it is interpreted as a range
                //The user can only enter in these three chars per the "AllowedSymbols" list therefore we don't
                //need to have default for the switch to fall through if no match case
                switch (SymbolToCheck)
                {
```

Extension Technique 3 – Regular expressions on the grid setting a cell to be part of a pattern

Regex is designed to pattern match. On its own it will not change any of the attributes of the cells within the grid once a pattern has been matched. To achieve this, we need to match patterns and then calculate where they are in the grid so that we can call appropriate methods on those cells. It is possible to recognise overlapping matches using regex; however, this limits the ability to then easily identify where those matches are. A solution to this is to iterate through the string representation of the grid looking for matches, although this increases the time complexity of the method. When matches are found, we need to record the position in the grid of those matches so that we can then call methods on the cells at those positions. This code improves the time complexity of the overall application from $O(n^3)$ to $O(n^2)$. You will need to add an accessor method and a mutator method to the Cell class to make this work. These should get and set a Boolean attribute called 'PartOfPattern'.

```
        private int CheckForMatchWithPatternRegexUsingGrid2(string SymbolToCheck)
        {
            string StringRepresentationOfWholeGrid = string.Join("", Grid.Select(cell => cell.GetSymbol()));
            Regex Check = new Regex("");
            switch (SymbolToCheck)
            {
                case "Q":
                    Check = new Regex($"Q{{2}}[Q|X|T|@|~]{{{GridSize - 2}}}Q{{2}}[Q|X|T|@|~]{{{GridSize}}}Q");
                    break;
                case "T":
                    Check = new Regex($"T{{3}}[Q|X|T|@|~]{{{GridSize - 2}}}T[Q|X|T|@|~]{{{GridSize - 2}}}T");
                    break;
                case "X":
                    Check = new Regex($"X[Q|X|T|@|~]X[Q|X|T|@|~]{{{GridSize - 3}}}[Q|X|T|@|~]X[Q|X|T|@|~][Q|X|T|@|~]{{{GridSize -
3}}}X[Q|X|T|@|~]X");
                    break;
            }
            var PatternMatches = new List<List<int>>();
            //This will match all of the cells in grid which contain the pattern, including the "spaces" in rows which I am not
interested in.
            //Regex doesn't match overlapping patterns so I need to iterate through the whole string version of the grid.
            for (int i = 0; i < StringRepresentationOfWholeGrid.Length - 2; i++)
```

```
        else if (!PatternMatches[PatternMatches.Count - 1].SequenceEqual(MatchPositions))
        {
            PatternMatches.Add(MatchPositions);
        }
    }
}

if (PatternMatches.Count > 0)
{
    //Included purely for testing
    Console.WriteLine("Symbol positions in each match in the grid:");
    foreach (var PatternMatch in PatternMatches)
    {
        //This may print multiple times if more than one pattern of the symbol being tested
        //currently exists in the grid.
        Console.WriteLine(string.Join(", ", PatternMatch));
    }

    bool ScoreAwarded = false;
    foreach (var PatternMatch in PatternMatches)
    {
        foreach (var CellPosition in PatternMatch)
        {
            //If there is a match, add the symbol to the SymbolsNotAllowedList for those cells.
            //But only if they are not already in a pattern.
            if (!Grid[CellPosition].IsPartOfPattern())
            {
                Grid[CellPosition].AddToNotAllowedSymbols(SymbolToCheck);
                Grid[CellPosition].SetPartOfPattern();
                ScoreAwarded = true;
            }
        }
    }
```

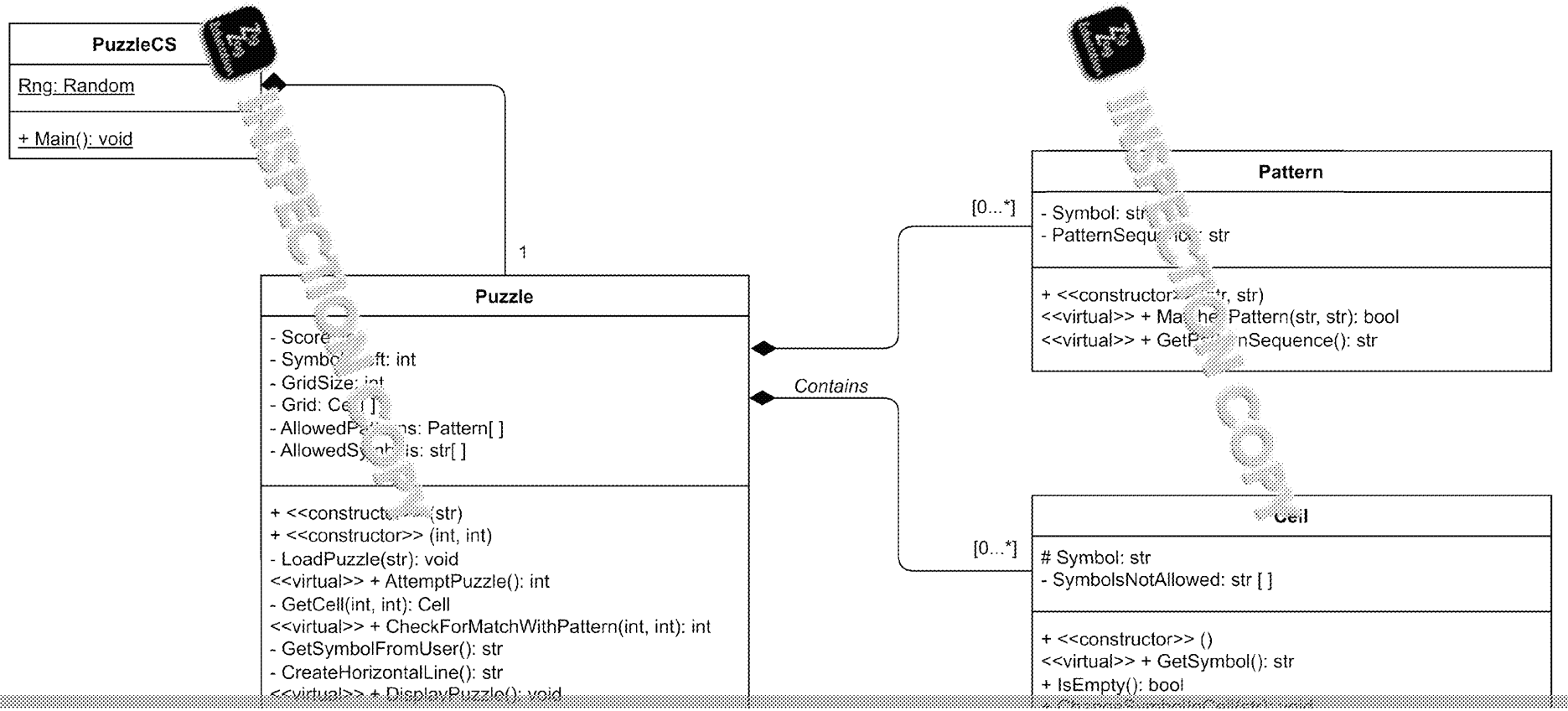
```
private List<int> GetJustMatchedSymbolPositionsInGrid(Match match)
{
    List<int> SymbolPositions = new List<int>() { 0,1,2 };
    for (int i = 2; i < GridSize*3; i++)
    {
        if (i >= GridSize - 1&& i <= GridSize + 2)
        {
            SymbolPositions.Add(i);

        }
        if (i >= (GridSize * 2) - 1 && i <= (GridSize * 2) + 2)
        {
            SymbolPositions.Add(i);
        }
    }
    return SymbolPositions.Select(i => match.Index + i)
                          .ToList();
}
```

Extension Technique 4 – Reverse referencing the grid

The skeleton code includes a method called GetCell which takes the parameters of Row and Column and returns the Cell at the associated location in the grid data structure. This method reverses that process, allowing the user to pass the reference of a location in the grid data structure and it returns a tuple containing the row and column for that cell.

```
private Tuple<int, int> GetRowColumnFromIndex(int Index)
{
    if (Index < 0 || Index >= Grid.Count)
    {
        return null;
    }
    int Result = Index / GridSize;
```

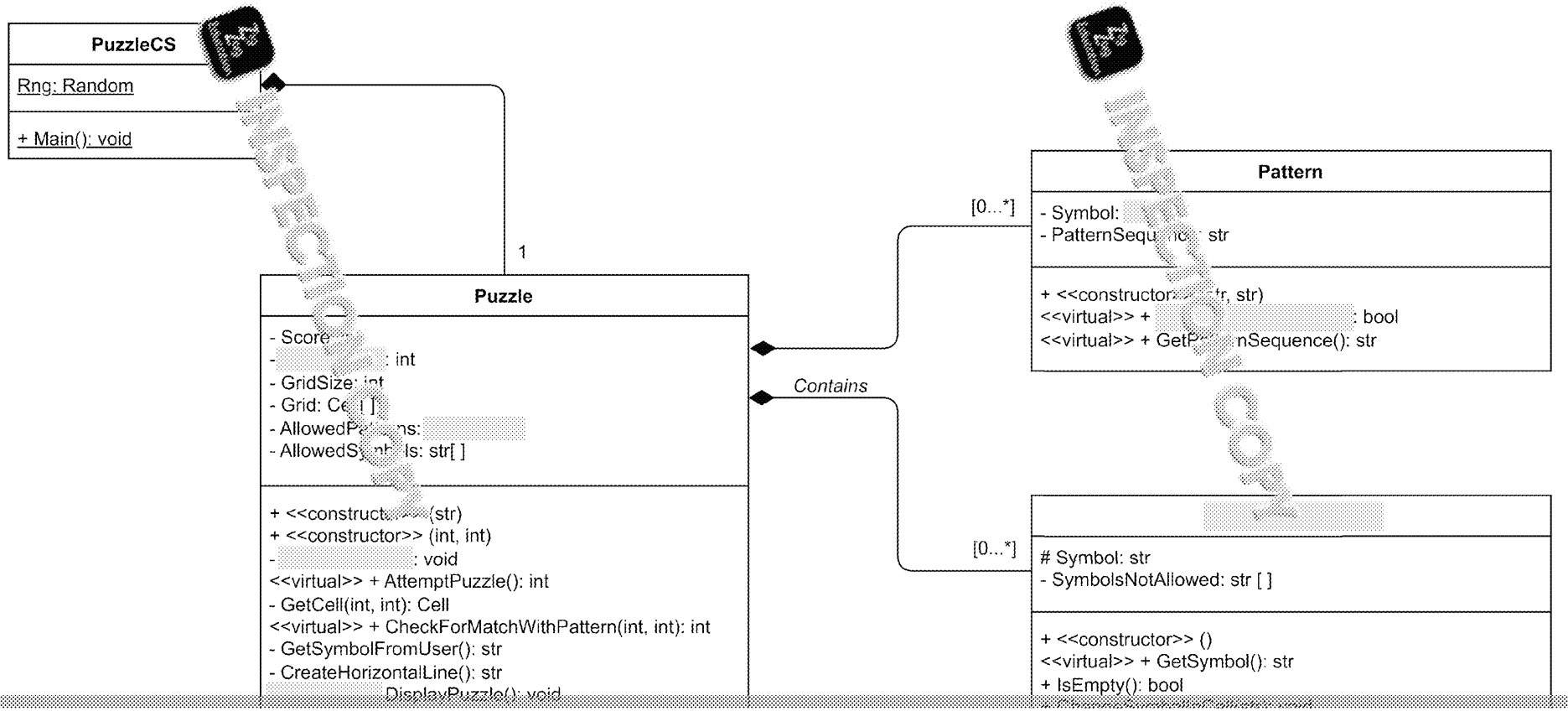
UML Class Diagram

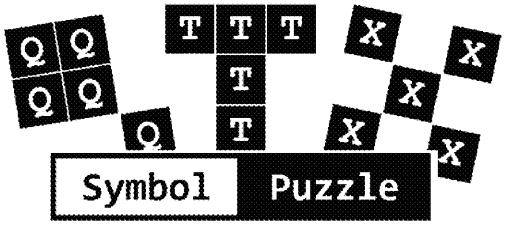


Symbol Puzzle

UML Class Diagram

Activity





Theory Questions

These questions are designed to test your understanding of the skeleton c to the kinds of question you can expect to see in Section C of the Paper 1 questions that are more than 2 marks are rarely seen in this section – thes here to challenge your understanding of the code.

These questions refer to the **Preliminary Material** and the **Sk** but **do not** require any additional programmin

TOTAL MARKS: 75

1 This question refers to the Main method in the PuzzleCS class and the

(a) Describe the purpose of the selection statement below, including parameters for the Puzzle constructor:

```
if (Filename.Length > 0)
{
  MyPuzzle = new Puzzle(Filename + ".txt");
}
else
{
  MyPuzzle = new Puzzle(8, Convert.ToInt32(
}
```

.....

.....

.....

.....

.....

.....

(b) Many languages, such as C# and Java, allow methods of the sam signatures to be defined in the same class.

State the name of this OOP technique.

.....

2 This question refers to the entire code. Throughout the code, several
hard-coded (see two examples below).

Example 1:

```
MyPuzzle = new Puzzle(8, Convert.ToInt32(8
```

Example 2:

```
if (Rng.Next(1, 101) < 90)
```

(a) State two reasons why constants would be more appropriate.

.....

.....

.....

(b) The code is writt[illegible] the object-oriented paradigm.

Di[illegible]s t[illegible]antages of this over the traditional structured app

.....

.....

.....

.....

.....

.....

3 This question refers to the CheckForMatchWithPattern method of the
pattern string used to match the Q pattern in the puzzle is: "QQ**Q**

(a) Explain how a successful match is determined.

.....

.....

.....

(b) Explain how an unsuccessful match is determined.

.....

.....

(c) E[illegible] h[illegible] players are prevented from placing any more letter Q
o[illegible] successful match has been determined.

.....

.....

15 This question involves the creation of a new algorithm. A new bonus s
for any player who manages to achieve the following 4 × 4 pattern in t

Q	Q	Q	Q
T	T	X	X
T	X	X	T
T	T	X	X

The bonus will be 50 points. Write an algorithm th can be used to c
algorithm can be in pseudocode, flowchart or s ctured English and s
not found and 50 if the pattern is fou in the grid. The algorithm sho
CheckForSpecialPattern. Y av sh to refer to the CheckForMatc
suitable logic.

END OF QUESTIONS

Symbol Puzzle

Theory Questions

These questions are designed to test your understanding of the skeleton c to the kinds of question you can expect to see in Section C of the Paper 1 questions that are more than 2 marks are rarely seen in this section – thes here to challenge your understanding of the code.

These questions refer to the **Preliminary Material** and the Sk but **do not** require any additional programmin

TOTAL MARKS: 75

1 This question refers to the Main method in the PuzzleCS class and the

(a) Describe the purpose of the selection statement below, including parameters for the Puzzle constructor:

```
if (Filename.Length > 0)
{
  MyPuzzle = new Puzzle(Filename + ".txt");
}
else
{
  MyPuzzle = new Puzzle(8, Convert.ToInt32(
}
```

(b) Many languages, such as C# and Java, allow methods of the sam signatures to be defined in the same class.

State the name of this OOP technique.

2 This question refers to the entire code. Throughout the code, several hard-coded (see two examples below).

Example 1:

```
MyPuzzle = new Puzzle(8, Convert.ToInt32(8
```

Example 2:

```
(Rng.Next(1, 101) < 90)
```

(a) State two reasons why constants would be more appropriate.

(b) The code is written using the object-oriented paradigm.

Discuss two advantages of this over the traditional structured app

13 This question refers to the DisplayPuzzle method of the Puzzle class. Th
which contains an iterative statement displays the column numbers and a

Describe in detail how the second iterative structure formats the rest o
on the Console.

Ensure that your description explains how the use of integer division a

14 This question refers to the constructor of the BlockedCell class.

(a) Explain the purpose of the keyword base().

(b) Describe the meaning of a public attribute.

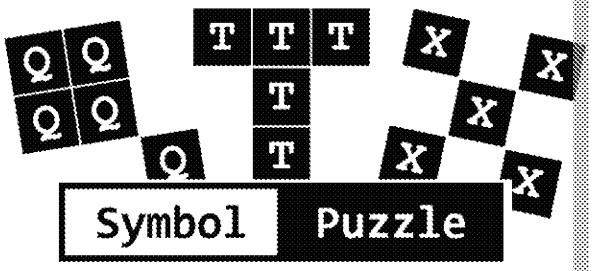
(c) Explain why the use of private and protecte[illegible]ibutes provides b
public attributes.

15 This question involves the creation of a new algorithm. A new bonus s
for any player who manages to achieve the following 4 × 4 pattern in t

	Q	Q	Q
T	T	X	X
T	X	X	T
T	T	X	X

The bonus will be 50 points. Write an algorithm that can be used to c
algorithm can be in pseudocode, flowchart or structured English and s
not found and 50 if the pattern is found in the grid. The algorithm sho
CheckForSpecialPattern. You may wish to refer to the CheckForMatc
suitable logic.

END OF QUESTIONS



Programming Tasks

These questions require you to l... t... Skeleton Program and to make

Note that any alternative or additional code changes that are deemed appropriate ... ring that it is clear where in the Skeleton Program those chang...

AQA has highlighted in the pre-release material that there are errors in the imp... Skeleton Program, meaning that it does not work as described under all circum... questions and answers in this document do not correct the errors in the implem... therefore, any anomalous output as a result of these errors will also be present i... this resource.

The objective of this resource is to provide you with a selection of different qu... questions. Questions which may have a similar theme may use different techni... options on how to solve problems. Some questions may contain a wider range of ... code than is realistic in an exam situation to stretch students – in particular Ques... are more prescriptive than others in how the task should be completed in order to ... solutions presented in this resource only use techniques which are included i...

Students are recommended to start with a clean copy of the pre-release code ... questions in this resource. This will prevent modifications made for one question ... different question.

Task 1

This question refers to the Puzzle class.

Introduce a new pattern option into the constructor of the Puzzle class for

What you need to do

Task 1.1

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

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

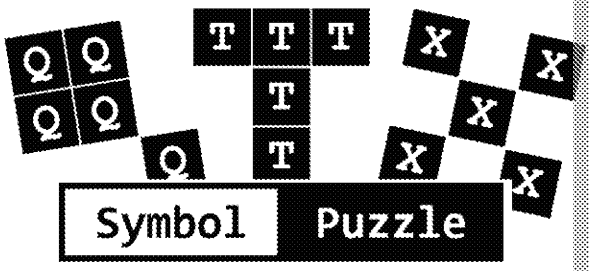
Task 1.2

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
  10 points.

Evidence that you need to provide:

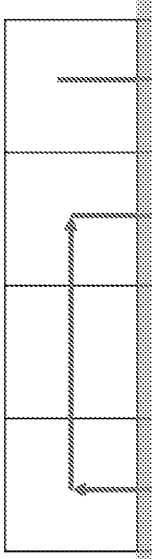
- Your PROGRAM SOURCE CODE showing the introduction of a new
  constructor of the Puzzle class.
- SCREEN CAPTURE(S) showing the required test.



Programming Tasks (Extensi

Extension 1

The current helix pattern sequ[illegible] epresentation of a 3 × 3 section of the gr[illegible] mits the range of letters which can be rep[illegible]te[illegible]n the space available for symbols. Introduce n[illegible]nctionality to allow the application to use a 4 × 4 section of the grid to represent a pattern sequence. Use this sequence to draw the helix pattern:



Extension 2

The application currently is a single-player puzzle. Introduce new functiona played by two players. A two-player puzzle should only be playable in a st Players should take it in turns to place symbols into the grid. Although both patterns and symbols (from a single SymbolsLeft total), player 1 places up player 2 places lower-case symbols. Players gain points by placing either pattern (for player 1) or a complete, valid lower-case pattern (for player 2). upper- and lower-case symbols is not valid. The ga[illegible]e[illegible]ds when the sym The winner is the player with the highest s[illegible]or[illegible].

Extensi[illegible]3

Currently th[illegible]er must decide where they would like to place a symbol to new functionality for the computer to 'suggest a move'. This costs the user offer the user the option to 'view a move suggestion', the computer should where a completed pattern (any of the symbols) can be placed into the gri user (option here for making the suggested option a different colour). Afte suggestion, the console is cleared, the user score is reduced by 5 and the state, allowing the user to then place symbols into the grid.

Preview of Questions Ends Here

Question		Answer	Mark	Guidance
10	a	A list is a dynamic data structure while an array is static // the size of a list can be changed during execution but the size of an array cannot [1]. A list can contain elements of various data types whereas an array has elements all of one data type [1].	1	MAX 1
10	b	An array is a static data structure whereas a list is dynamic so it will be stored as a single contiguous block of memory [1], making it more time efficient [1]. Using two dimensions is more intuitive as it matches the structure shown to the user [1] whereas a one-dimensional list needs a conversion from the row and column input to give the cell [1]. The two-dimensional array will pick up input errors correctly [1] whereas the one-dimensional list will not always as it will frequently just return a different element [1].	4	MAX 4
11	a	All possible 3 × 3 sections around the square upon which the player has placed a symbol are checked (using a nested for loop). // For each 3 × 3 grid, a string is collected in a helix shape to generate a string to be used for comparison. // If a match is found then the function/method returns 10 which stops any further matches from being detected [1].	1	
11	b	(3,3) (3,5) (1,1) (1,5) OR (3,3) (3,5) (1,5) (1,1) OR (3,3) (1,1) (3,5) (1,5)	2	Use whichever solution allows most matches from the start. Award 1 mark for two correct in order (stop when wrong match) and 2 marks for all four correct.
12	a	O(n!)	1	
12	b	This is a problem in which all combinations need to be attempted [1]. This means factorial time complexity as when a new possibility is added it must be tried with all existing possibilities [1]. It is the same problem/concept/idea as the travelling salesman problem [1].	2	MAX 2
12	c	A tractable problem has a solution [1] in polynomial time or better [1]	2	
13		The second iterative statement loops through every cell in the grid [1]. For each cell: • If it's the start of the line [1] and the GridSize is less than 10 [1] then write out the row number and a space [1] • Write out a vertical bar and the symbol in the current Cell [1]	6	

Task 17 (11 marks)

Coding

- Creation of a WildCard class. *[1 mark]*
- Set the wild card symbol to ‘W’ and include suitable accessor methods to identify it as being a wild card. *[1 mark]*
- Required virtual methods in the Cell class to make the WildCard class operate correctly. *[1 mark]*
- Suitable variable to count the number of wild cards available. *[1 mark]*
- Advise the user of how many wild cards they have available to use. *[1 mark]*
- Test to confirm if a wild card needs to be placed when inputting a new symbol into the grid. *[1 mark]*
- Insert a wild card into a valid location in the grid and update the number of wild cards available. *[1 mark]*
- When a pattern helix is created, check if any of the cells within the pattern helix are a wild card. *[1 mark]*
- Suitable string handling to replace any wild card symbols in the pattern helix with the appropriate pattern symbol to test against. *[1 mark]*
- If a match is made when a wild card is present, return 15 points. *[1 mark]*

Example Solution

Creation of new WildCard class:

```
//CHANGE
class WildCardCell : Cell
{
    public WildCardCell() : base()
    {
        Symbol = "W";
    }
    public override bool IsWild()
    {
        return true;
    }
}
//END CHANGE
```

Modification to the Pattern class to get the pattern symbol:

```
//CHANGE
public string GetPatternSymbol()
{
    return Symbol;
}
//END CHANGE
```

Modification to the CheckForMatchWithPattern method to additionally allow wild cards to be placed:

```
public virtual int CheckForMatchWithPattern(int Row, int Column)
{
    for (var StartRow = Row + 2; StartRow >= Row; StartRow--)
    {
        for (var StartColumn = Column - 2; StartColumn <= Column; StartColumn++)
        {
            try
            {
                //CHANGE
                foreach (var P in AllowedPatterns)
                {
                    string PatternString = "";
                    List<Cell> PatternCells = new List<Cell>();
                    PatternCells.Add(GetCell(StartRow, StartColumn));
                    PatternCells.Add(GetCell(StartRow, StartColumn + 1));
                    PatternCells.Add(GetCell(StartRow, StartColumn + 2));
                    PatternCells.Add(GetCell(StartRow - 1, StartColumn + 2));
                    PatternCells.Add(GetCell(StartRow - 2, StartColumn + 2));
                    PatternCells.Add(GetCell(StartRow - 2, StartColumn + 1));
                    PatternCells.Add(GetCell(StartRow - 2, StartColumn));
                    PatternCells.Add(GetCell(StartRow - 1, StartColumn));
                    PatternCells.Add(GetCell(StartRow - 1, StartColumn + 1));
```

```
                    if (P.MatchesPattern(PatternString, CurrentSymbol))
                    {
                        GetCell(StartRow, StartColumn).AddToNotAllowedSymbols(CurrentSymbol);
                        GetCell(StartRow, StartColumn + 1).AddToNotAllowedSymbols(CurrentSymbol);
                        GetCell(StartRow, StartColumn + 2).AddToNotAllowedSymbols(CurrentSymbol);
                        GetCell(StartRow - 1, StartColumn + 2).AddToNotAllowedSymbols(CurrentSymbol);
                        GetCell(StartRow - 2, StartColumn + 2).AddToNotAllowedSymbols(CurrentSymbol);
                        GetCell(StartRow - 2, StartColumn + 1).AddToNotAllowedSymbols(CurrentSymbol);
                        GetCell(StartRow - 2, StartColumn).AddToNotAllowedSymbols(CurrentSymbol);
                        GetCell(StartRow - 1, StartColumn).AddToNotAllowedSymbols(CurrentSymbol);
                        GetCell(StartRow - 1, StartColumn + 1).AddToNotAllowedSymbols(CurrentSymbol);
                        if (WildCardInPattern)
                        {
                            return 15;
                        }
                        else
                        {
                            return 10;
                        }
                    }
                    //END CHANGE
                }
                catch
                {
                }
            }
        }
        return 0;
    }
```

Suitable variable to count the number of wild cards available:

Modification to the AttemptPuzzle method:

```
        public virtual int AttemptPuzzle()
        {
            bool Finished = false;
            while (!Finished)
            {
                DisplayPuzzle();
                Console.WriteLine("Current score: " + Score);
                //CHANGE
                Console.WriteLine("Wild cards will allow two symbols to be placed on top of each other and still be matched for
each pattern.");
                Console.WriteLine("You currently have " + WildCardsLeft + " wild cards left.");
                //END CHANGE
                bool Valid = false;
                int Row = -1;
                while (!Valid)
```

…

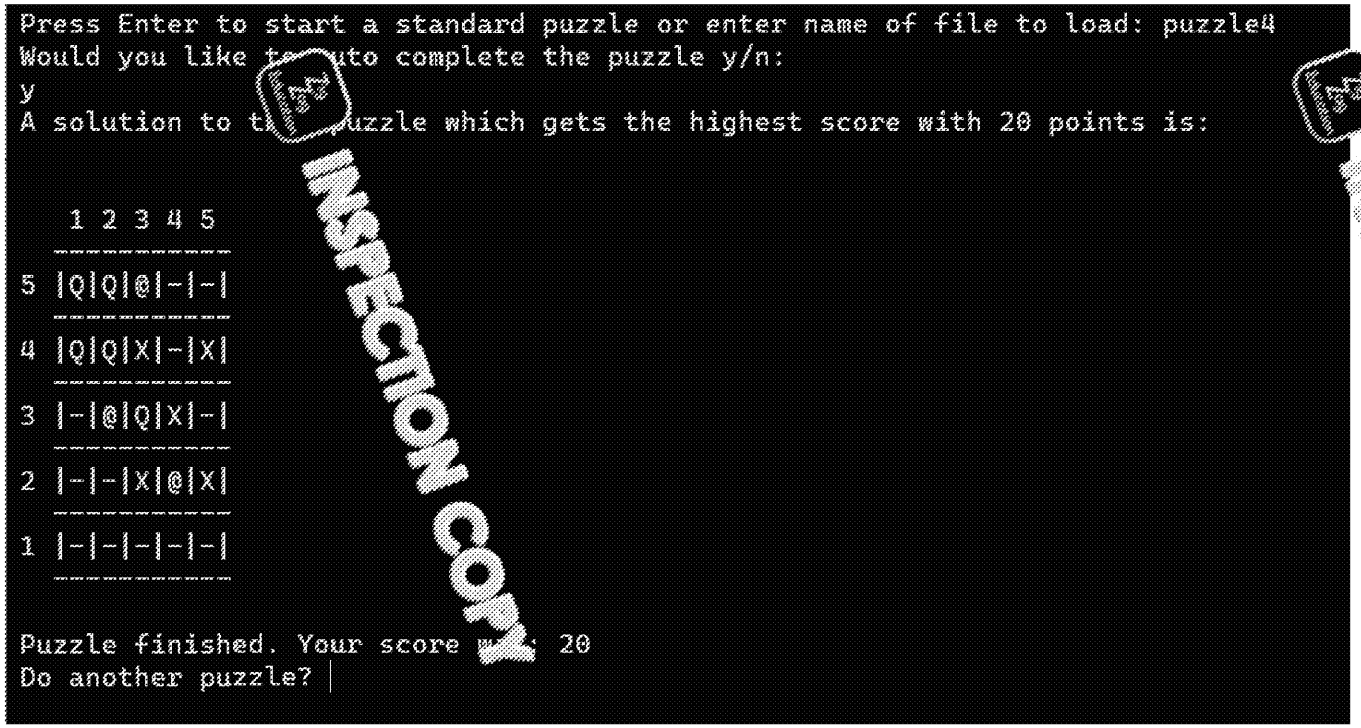
```
                Cell CurrentCell = GetCell(Row, Column);
                if (CurrentCell.CheckSymbolAllowed(Symbol))
                {
                    //CHANGE
                    if (!CurrentCell.IsEmpty() && CurrentCell.GetSymbol() != "@")
                    {
                        if (WildCardsLeft > 0)
                        {
                            Grid[(GridSize - Row) * GridSize + Column - 1] = new WildCardCell();
                            WildCardsLeft--;
                        }
                        else
                        {
                            Console.WriteLine("You have used both your wildcards for this puzzle.");
                            continue;
                        }
```

Testing

- Displaying an overlapping T pattern using a wild card. *[1 mark]*

Testing

- Displaying an auto completed puzzle4 file with the following layout and score of 20. *[1 mark]*



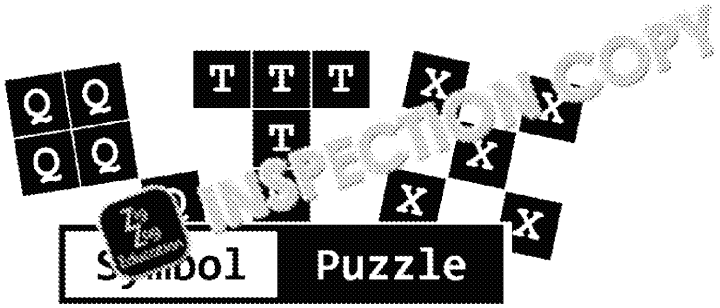
Preview of Answers Ends Here

Name	

ZigZag Education supporting

A Level AQA Computer Science Pap

Summer 2024



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:

Exam-style Questions

Answer all questions. Remember to save this document

Q		Answer
1	(a)	
	(b)	
2	(a)	
	(b)	
3	(a)	
	(b)	
	(c)	
4		
5	(a)	
	(b)	
	(c)	
6	(a)	
	(b)	
7	(a)	
	(b)	
8	(a)	
	(b)	
9	(a)	
	(b)	
10	(a)	
	(b)	
11	(a)	
	(b)	
12	(a)	
	(b)	
	(c)	
13		
14	(a)	
	(b)	
	(c)	
15		

Exam-style Programming Task

Answer all questions. Remember to save this document

Q	Answer
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	