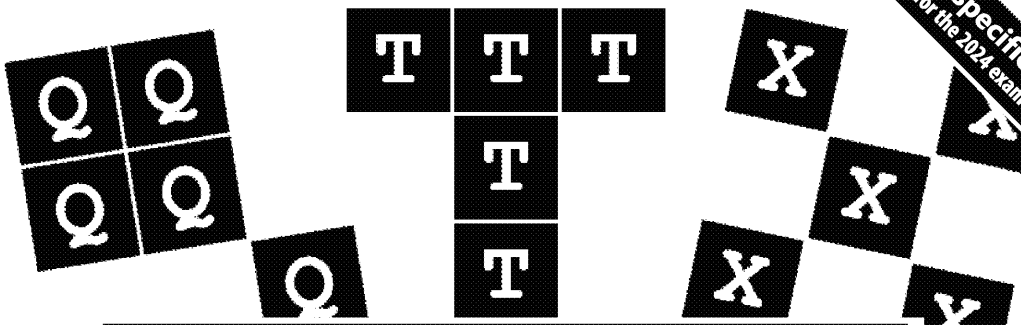
Zig Zag Education

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

PYTHON³ EDITION

- DIGITAL RESOURCE -

This pack includes paper versions of the electronic files.

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

POD 12195

zigzageducation.co.uk

Contents

Product Support from ZigZag Education ii

Terms and Conditions of Use iii

Teacher's Introduction iv

Printouts of electronic resources (for reference)

- Code Breakdown (7 pages)
- Puzzle File Breakdown (3 pages)
- Advanced Techniques (5 pages)
- UML Class Diagram – Complete (1 page)*
- Theory Questions: Write-on Version (10 pages)
- Theory Questions: Non-write-on Version (5 pages)
- Coding Tasks (21 pages)
- Additional Tasks (Extension) (3 pages)
- Theory Questions: Mark Scheme (4 pages)
- Programming Tasks: Mark Scheme (58 pages)
- Electronic Answer Document (3 pages)

** 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.

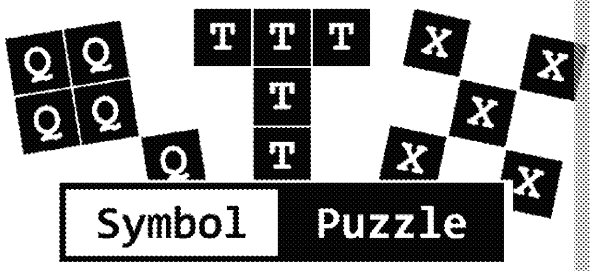
Installation: Extract the files from the downloaded ZIP file and move the entire `SymbolPuzzle` folder onto a network location that is accessible for students, and provide them with a shortcut to the index.html file. All content can be accessed from this page.

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:

- `py02a-UML-Diagam-Complete.pdf`
- `py06-TheoryQuestions-MS.pdf`
- `py07-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:	This is the main entrance poi to determine whether the app standard, randomly created p puzzle text file. It initialises a string variable c value 'y' and an integer varia The method then enters into held in that loop by the value operates using the following s • Prompt the user if they w puzzle or load an externa • Instantiate a new Puzzle • If the user has entered a constructor in the Puzzle as a parameter. This will external text file. • If the user does not enter by the length of the filena overloaded constructor in the values 8 and 38. The for a standard puzzle, cre wide by 8 rows. The seco squaring the GridSize, m then rounding down to an • The method then calls th Puzzle which starts th When a puzzle is comple the Score variable, whic • The method then prompts do another puzzle, settin appropriately to either st the program, or allow it t
Return val: n/a	

Class: Puzzle

Identifier / Data		Description
<<constructor>>		
Parameters	Filename : String OR Size : Int StartSymbols : Int	Uses overloading to accept tw constructor. **This version of the construc external puzzle file is loade** If one string argument is pass loading an external puzzle file Initialises the following private • Score to 0 • SymbolsLeft from param • GridSize from parameter • Grid as List • AllowedPatterns as List • AllowedSymbols as List These attributes are subsequ LoadPuzzle() method. The method calls the LoadPu Filename parameter. **This version of the construc standard puzzle is generate** If two integer arguments are p creating a standard puzzle), th Initialises the following private • Score to 0 • SymbolsLeft from param • GridSize from parameter • Grid as List The method performs a count the square of the GridSize. It creates and adds all the cells Grid. Using a random numbe has a 90% chance of being a otherwise a blocked cell. The method then initialises A patterns and AllowedSymbo generates the default pattern patterns, adding them to the A with adding the associated sy AllowedSymbols list.
Return values	n/a	

AttemptPuzzle (public)	
Parameters	n/a
Return values	Score : Int

This method is the main loop
time. It is held in the loop usin
Finished.

The method firstly displays th
calling the DisplayPuzzle() m
current user Score.

The method then asks the us
and Column of where they w
the pu The method uses
e er has entered intege
ey are valid within the boun
not give any error messages

The method then calls the Ge
to get a symbol to place into t
decrements the number of sy
in the puzzle.

The method then creates a lo
location given by the user an
given by the user can be use
CheckSymbolAllowed() me
entered by the user as a para
used in that cell, the symbol i
ChangeSymbolInCell() meth

The method then checks if th
match by calling the Checkfo
passing Row and Column en
parameters. The result of this
integer variable AmountToA
than 0, it is added to the user

The method then checks if all
been used, and if so, exits the

If the main program has exite
displayed by calling the Displ
user Sco is returned.

CheckforMatchWithPattern (public)		
Parameters	Row : Int Column : Int	Uses a nested loop to iterate th together a string variable calle built from nine cells in a 3 × 3 a helix. A nested loop is used combinations of pattern that c different locations in a 3 × 3 se with a try…catch structure to p a cell location outside of the b Once the PatternString has uses a foreach loop to check Allowed Patterns list by calli method, passing the Pattern checked as parameters. If a match is found, the metho AddToNotAllowedSymbols matching 3 × 3 section of the being checked as a paramete symbol from being placed int If a match has been found, th 10, otherwise it returns the S
Return values	Int	
CreateHorizontalLine (private)		
Parameters	n/a	Uses iteration to concatenate characters which is the corre in the current puzzle.
Return values	Symbol : String	
DisplayPuzzle (public)		
Parameters	n/a	Used to print out the grid onto works by using the following s • Print a blank line. • If the GridSize is less tha the screen, then iterate th a space followed by the c • Print a blank line. • Print a horizontal line by CreateHorizontalLine() • The method then iterates printing out the row num symbol and the symbol i iteration uses the MOD f a row before printing a horizontal line underneat • This process is repeated printed to the screen.
Return values	n/a	
GetCell (		
Parameters	Row : Int Column : Int	Uses the Row and Column p correct Cell element in the or is then returned. If the Row a generate an index location w raises an IndexError. *The ot do not do this.*
Return values	Cell	

GetSymbolFromUser (private)		
Parameters	n/a	Used for getting a symbol fro grid. The method uses a loop enter the symbol they want to symbol which is part of the A When they enter a valid symb
Return values	Symbol : String	

LoadPuzzle (private)		
Parameters	Filename : String	The method loads an externa parameter. See 'P z File Breakdown' does an external puzzle file The method sequences throu from the Filename parameter Assigns NoOfSymbols from uses this value to iterate thro the symbols from those lines Assigns NoOfPatterns from uses this value to iterate thro in each pattern. A pattern line list – the first element is the s second element is the pattern instantiate a new pattern obje AllowedPatterns list. Assigns GridSize from the ne the square of this value to ite reading in each cell. A cell lin list – the first element is the s remaining elements is a subli symbols for the SymbolsNot first element is an '@' symbo and appends a BlockedCell application instantiates a Cel subsequent symbols not allow Once all the cells have been method assigns the next line and the final line to the Symb Th method uses a try…catch an error occurs, an error me the method does not give the reload the file.
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)		
Parameters	n/a	Returns the value of the private
Return values	PatternSequence : String	
MatchesPattern (public)		
Parameters	Patt[illegible] String [illegible]Placed : String	This is used to confirm that a pa concatenation through a 3 × 3 s PatternSequence attribute in a
Return va[illegible]	Boolean	If the passed parameter Symbo Symbol for the pattern, the met 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)		
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)		
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)		
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	

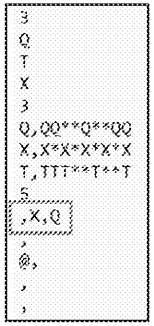
Puzzle File Breakdown of Puzzle1.txt

Line Number	Data	Description
1	3	This is the number of symbols in the game. It is used to iterate through the next thr
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 eleme the pattern, and the second element is the PatternSequence. The PatternSequence represents the pattern as a helix of c 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 calcu 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 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. W it is put into the SymbolsNotAllowed list, which implies multiple symbols can be put into the list. If a line contains this seco it has been matched and scored as part of a pattern in the puzzle.
12	@,Q	The @ says that this is a blocked cell. In this example, however, the blocked cell is within th 3 section of a matched because it contains a SymbolsNotAllowed list. While a blocked cell can have this att e, it serves no purpose in a block
13	,	This is a simple empty cell.
14	,	
15	Q,Q	
16	Q,Q	

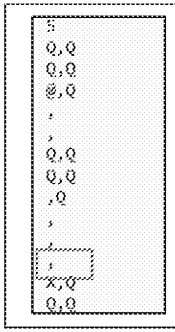
Line Number	Data	Description
17	,Q	This is a blank cell. In this example, however, the cell is within the 3 × 3 section of a matched pattern in the puzzle becaus SymbolsNotAllowed list.
18	,	
19	,	
20	,	See the 'Error' description before this row.
21	X,Q	This cell has an X symbol in it; however, the cell is within the 3 × 3 section of a matched pattern in the puzzle because it c SymbolsNotAllowed list. This is because the Q pattern was created before the X pattern; therefore, Q was used as the 'S 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 be therefore, there are currently no symbols banned from this cell.
28	,	
29	,	
30	,	
31	X	
32	,	
33	,	
34	,	
35	10	This is the score of the current game.
36	1	This is the number of symbols left than can be played in this game.

‘Errors’ in the puzzle files

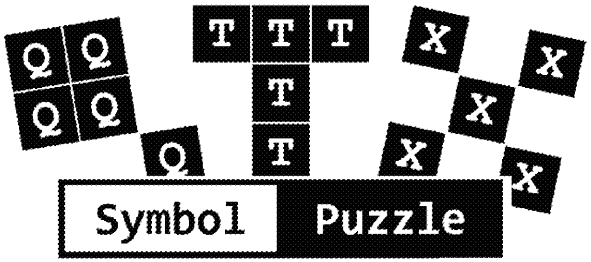
There are two logical errors that we have found in the puzzle files. These ‘errors’ have been checked with AQA, who have confirmed that the fil expect them to or as required for the purpose of the exam. While they don’t impact the main functionality of the application, the ‘errors’ generate behaviour 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 solving challenges presented for the pre-release material. The objective of this document is to extend students' knowledge and explore alternat could use as they experiment with the code. **Use of these techniques and ideas should be at the teacher's discretion**. The code shown be 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 incl question which uses regex. *(A regex question in Section C is perfectly possible.)* Although the library has not been included, students can impo 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: `import re`

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

```
    def CheckforMatchWithPatternRegexAQAStandard(self, PatternStringToCheck):
        # These expressions comply with the techniques in the AQA specification
        # The hyphen must be last otherwise it is interpreted as a range
        t_check = re.compile(r'TTT[Q|X|T|@|-][Q|X|T|@|-]T[Q|X|T|@|-][Q|X|T|@|-]')
        x_check = re.compile(r'X[Q|X|T|@|-]X[Q|X|T|@|-]X[Q|X|T|@|-]X[Q|X|T|@|-]X')
        q_check = re.compile(r'QQ[Q|X|T|@|-][Q|X|T|@|-]Q[Q|X|T|@|-][Q|X|T|@|-]QQ')

        return bool(t_check.search(PatternStringToCheck)) or bool(x_check.search(PatternStringToCheck)) or bool(q_c
ernStringToCheck))
```

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

```
    def CheckforMatchWithPatternRegexBeyondStandard(self, PatternStringToCheck):
        # These expressions use techniques which are beyond the AQA specification
        # The hyphen must be last otherwise it is interpreted as a range
        t_check = re.compile(r'T{3}[Q|X|T|@|-]{2}T[Q|X|T|@|-]{2}T')
        x_check = re.compile(r'(X[Q|X|T|@|-]){4}X')
        q_check = re.compile(r'Q{2}[Q|X|T|@|-]{2}Q[Q|X|T|@|-]{2}QQ')

        return bool(t_check.search(PatternStringToCheck)) or bool(x_check.search(Patt      ringToCheck)) or bool(q_c
ernStringToCheck))
```

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 main data structure for material uses a one-dimensional list; therefore, regex combined with a lambda-like expression could be used on this to make the same improve the need to generate the helix pattern string. The expression, however, needs further modification to take into account that the grid can be diffe therefore, that the space between symbols within a pattern can change. A limitation of this code, however, is that it cannot detect overlapping p patterns right next to each other because the regex is not identifying overlapping matches.

```
    def CheckforMatchWithPatternRegexUsingGrid(self, SymbolToCheck):
        StringRepresentationOfWholeGrid = ''.join(cell.GetSymbol() for cell in self.__Grid)
        check = None

        # 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 a default for the switch to fall through if no match case
        if (SymbolToCheck == "Q"):
            check = re.compile(f'Q{{2}}[Q|X|T|@|-]{{{self.__GridSize - 2}}}Q{{2}}[Q|X|T|@|-]{{{self.__GridSize}}}Q')
        elif (SymbolToCheck == "T"):
            check = re.compile(f'T{{3}}[Q|X|T|@|-]{{{self.__GridSize - 2}}}T[Q|X|T|@|-]{{{self.__GridSize - 1}}}T')
        elif (SymbolToCheck == "X"):
            check = re.compile(f'X[Q|X|T|@|-]X[Q|X|T|@|-]{{{self.__GridSize - 3}}}[Q|X|T|@|-]X[Q|X|T|@|-][Q|X|T|@|-
]{{{self.__GridSize - 3}}}X[Q|X|T|@|-]X')

        return bool(check.search(StringRepresentationOfWholeGrid))
```

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 match 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 poss overlapping matches using regex; however, this limits the ability to then easily identify where those matches are. A solution to this is to iterate t representation of the grid looking for matches, although this increases the time complexity of the method. Where matches are found, we need t 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 o 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 sh Boolean attribute called 'PartOfPattern'.

```
    def CheckforMatchWithPatternRegexUsingGrid2(self, SymbolToCheck):
        StringRepresentationOfWholeGrid = ''.join(cell.GetSymbol() for cell i      l(),__Grid)
        check = None
        pattern_matches = []

        # 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 a default for the switch to fall through if no match case
        if (SymbolToCheck == "Q"):
            check = re.compile(f'Q{{2}}[Q|X|T|@|-]{{{self.__GridSize - 2}}}Q{{2}}[Q|X|T|@|-]{{{self.__GridSize}}}Q')
        elif (SymbolToCheck == "T"):
            check = re.compile(f'T{{3}}[Q|X|T|@|-]{{{self.__GridSize - 2}}}T[Q|X|T|@|-]{{{self.__GridSize - 1}}}T')
        elif (SymbolToCheck == "X"):
            check = re.compile(f'X[Q|X|T|@|-]X[Q|X|T|@|-]{{{self.__GridSize - 3}}}[Q|X|T|@|-]X[Q|X|T|@|-][Q|X|T|@|-
]{{{self.__GridSize - 3}}}X[Q|X|T|@|-]X')

        # This will match all of the cells in the grid which contain the pattern, including the "spaces" in rows wh
erested in.
        # Regex doesn't match overlapping patterns, so I need to iterate through the whole string version of the gr
        for i in range(len(StringRepresentationOfWholeGrid) - 9):
            match = check.search(StringRepresentationOfWholeGrid, i)
            if (match):
                match_positions = self.GetJustMatchedSymbolPositionsInGrid(match)
                if (not pattern_matches or pattern_matches[-1] != match_positions):
                    pattern_matches.append(match_positions)
```

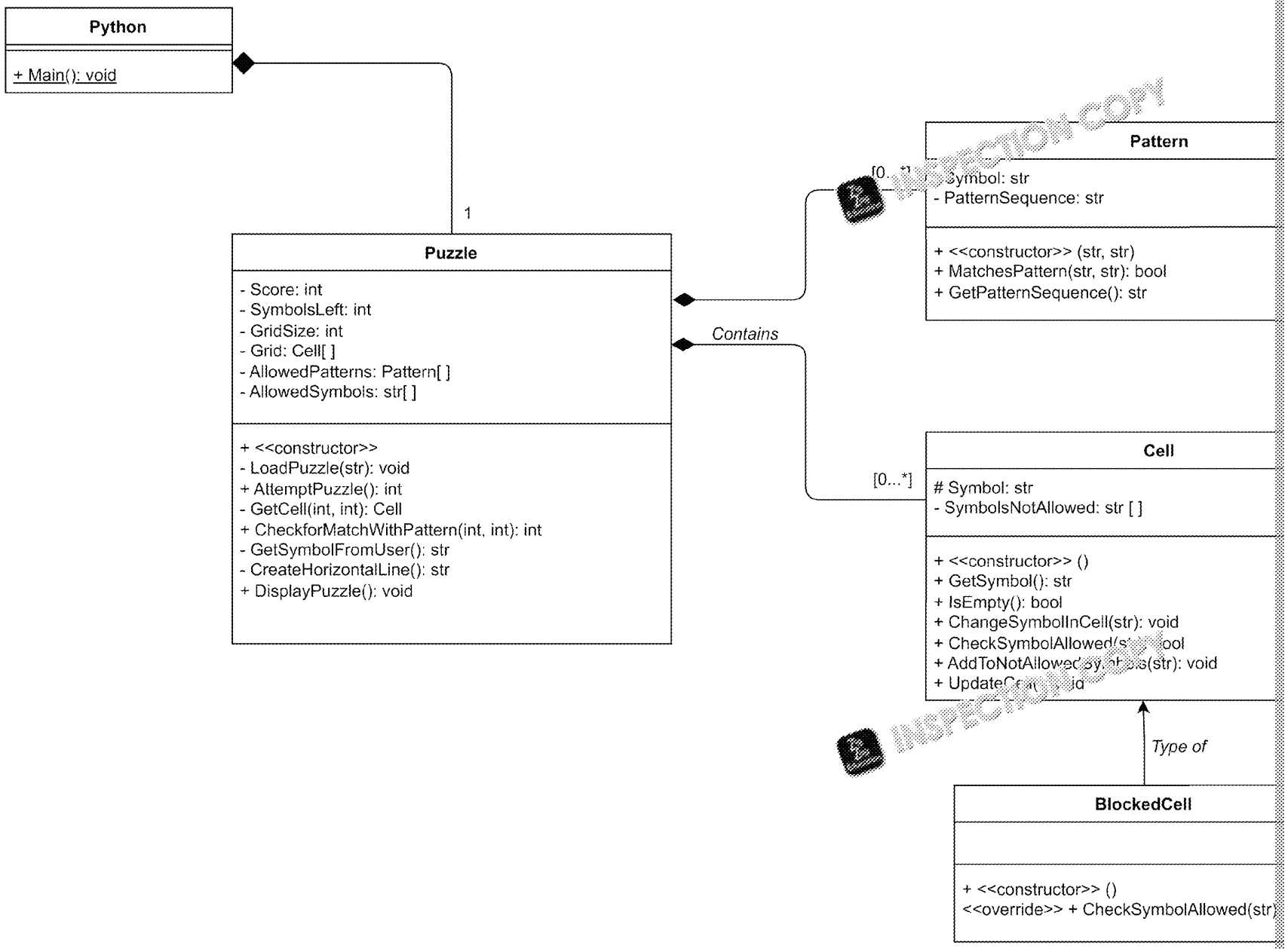
```
        if (pattern_matches):
            # Included purely for testing
            print("Symbol positions in each match in the grid:")
            for pattern_match in pattern_matches:
                # This may print multiple times if more than one pattern of the symbol being tested
                # currently exists in the grid.
                print(", ".join(map(str, pattern_match)))

            score_awarded = False
            for pattern_match in pattern_matches:
                for cell_position in pattern_match:
                    # 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 not self.__Grid[cell_position].IsPartOfPattern():
                        self.__Grid[cell_position].AddToNotAllowedSymbols(SymbolToCheck)
                        self.__Grid[cell_position].SetPartOfPattern()
                        score_awarded = True

            if (score_awarded):
                return 10
        else:
            print(f"No pattern matches for Symbol: {SymbolToCheck} found.")

        return 0

    def GetJustMatchedSymbolPositionsInGrid(self, match):
        symbol_positions = [0, 1, 2]

        for i in range(2, self.__GridSize * 3):
            if self.__GridSize - 1 < i <= self.__GridSize + 2:
                symbol_positions.append(i)
            if (self.__GridSize * 2) - 1 < i <= (self.__GridSize * 2) + 2:
                symbol_positions.append(i)

        return [i + match.start() for i in symbol_positions]
```

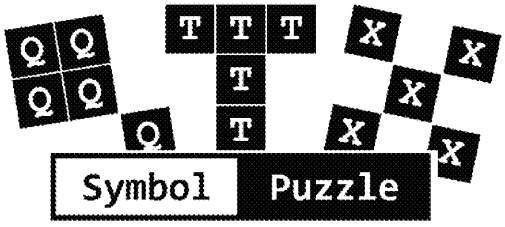
Extension Technique 4 – Reverse referencing the grid

The skeleton code includes a method called GetCell which takes the parameters of Row and Column and returns the Cell at the associated loc
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
the row and column for that cell.

```
    def GetRowColumnFromIndex(self, Index):
        if (Index < 0 or Index >= len(self.__Grid)):
            return None
        Result = Index // self.__GridSize
        row = (self.__GridSize - 1) - Result
        column = Index % self.__GridSize
        return row + 1, column + 1
```

UML Cla





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 subroutine and the constructor of the

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

```
if len(Filename) > 0:
    MyPuzzle = Puzzle(Filename + ".txt")
else:
    MyPuzzle = Puzzle(8, int(8 * 8 * 0.6))
```

.....

.....

.....

.....

.....

.....

(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 = Puzzle(8, int(8 * 8 * 0.6))
```

Example 2:

```
if random.randrange(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] t[illegible]vantages of this over the traditional structured app

.....

.....

.....

.....

.....

.....

3 This question refers to the CheckforMatchWithPattern method of the P
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.

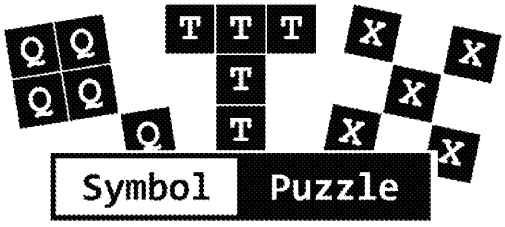
.....

.....

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

.....

.....



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 subroutine and the constructor of the

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

```
if len(Filename) > 0:
  MyPuzzle = Puzzle(Filename + ".txt")
else:
  MyPuzzle = Puzzle(8, int(8 * 8 * 0.6))
```

(b) Many languages, such as C# and Java, allow methods of the san 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 = Puzzle(8, int(8 * 8 * 0.6))
```

Example 2:

```
if random.randrange(1, 101) < 90:
```

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

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

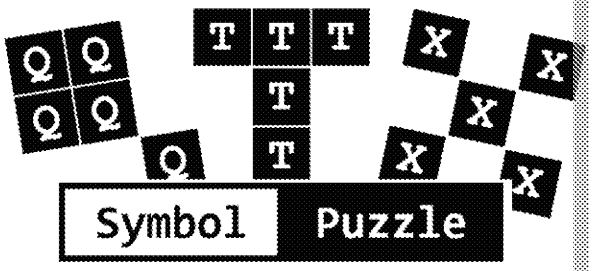
Discuss two advantages of this over the traditional structured app

3 This question refers to the CheckforMatchWithPattern method of the P 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) Explain how players are prevented from placing any more letter Q once a successful match has been determined.



Programming Tasks

These questions require you to load the **Skeleton Program** and to make

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

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:

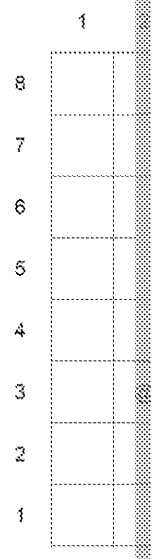
- Run the Skeleton Program.
- Press `enter` to start a standard puzzle.
- Input the new pattern into an available space on the grid.
- Show the program displaying the new pattern in a standard puzzle and 10 points.

Evidence that you need to provide:

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

Task 2

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.



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

The program should still allow the user to place symbols into the grid in the 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 ide helix, which is not blocked, has already been used in a pattern and, if so, r

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 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` *(These grid locations may need adjusting to fit the space available i*
- Show the program displaying the puzzle with overlapping T patterns a

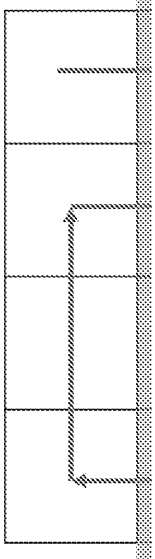
Evidence you need to provide:

- Your PROGRAM SOURCE CODE for the amended method CheckforMatchWithPattern in the Puzzle class.
- SCREEN CAPTURE(S) showing the required test.

Programming Tasks (Extensi

Extension 1

The current helix pattern sequence is a representation of a 3 × 3 section of the grid. This limits the range of letters which can be represented with the space available for symbols. Introduce new functionality 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 a pattern (for player 1) or a complete, valid lower-case pattern (for player 2). upper- and lower-case symbols is not valid. The game ends when the sym The winner is the player with the highest score.

Extens 3

Currently the user 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). After 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	Gu
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 index 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 [1]. If a match is found then the function/method returns 10 which stops any further matches from being detected [1].	2	
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 s matches from th mark for two cor when wrong ma 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] • If it's the end of the line [1] then it writes a vertical bar and then a horizontal line on the line below and repeats [1]	6	
14	a	super() is used to call a method in the parent of a class	1	
14	b	A public attribute can be accessed by anyone / anything / any class	1	
14	c	Private and protected attributes mean that getter/setter (accessor) methods have to be used [1]. These methods control access to the attributes [1] and allow validation/formatting/checks to be put in place [1] and for the underlying implementation to actually differ from the interface if necessary [1]. There is no direct access to the property outside of the class when using private and protected attributes. [1]	3	MAX 3

Task 17

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):
    def __init__(self):
        super(WildCardCell, self).__init__()
        self._Symbol = "W"

    def IsWild(self):
        return True

#END CHANGE
```

Modification of the Cell class to make the `WildCard` class operate correctly:

```
    #CHANGE
    def IsWild(self):
        return False
    #END CHANGE
```


Modification to the `Pattern` class to get the pattern symbol:

```
    #CHANGE
    def GetPatternSymbol(self):
        return self.__Symbol
    #END CHANGE
```

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

```
    def CheckforMatchWithPattern(self, Row, Column):
        for StartRow in range(Row + 2, Row - 1, -1):
            for StartColumn in range(Column - 2, Column + 1):
                try:
                    #CHANGE
                    for P in self.__AllowedPatterns:
                        PatternString = ""
                        PatternCells = []
                        PatternCells.append(self.__GetCell(StartRow, StartColumn))
                        PatternCells.append(self.__GetCell(StartRow, StartColumn + 1))
                        PatternCells.append(self.__GetCell(StartRow, StartColumn + 2))
                        PatternCells.append(self.__GetCell(StartRow - 1, StartColumn + 2))
                        PatternCells.append(self.__GetCell(StartRow - 2, StartColumn + 2))
                        PatternCells.append(self.__GetCell(StartRow - 2, StartColumn + 1))
                        PatternCells.append(self.__GetCell(StartRow - 2, StartColumn))
                        PatternCells.append(self.__GetCell(StartRow - 1, StartColumn))
                        PatternCells.append(self.__GetCell(StartRow - 1, StartColumn + 1))
                        WildCardInPattern = False
                        for C in PatternCells:
                            if (C.IsWild()):
                                WildCardInPattern = True
                            PatternString += C.GetSymbol()
                        if (WildCardInPattern):
                            PatternString = PatternString.replace("W", P.GetPatternSymbol())
                        CurrentSymbol = self.__GetCell(Row, Column).GetSymbol()
                        if P.MatchesPattern(PatternString, CurrentSymbol):
                            self.__GetCell(StartRow, StartColumn).AddToNotAllowedSymbols(CurrentSymbol)
                            self.__GetCell(StartRow, StartColumn + 1).AddToNotAllowedSymbols(CurrentSymbol)
                            self.__GetCell(StartRow, StartColumn + 2).AddToNotAllowedSymbols(CurrentSymbol)
                            self.__GetCell(StartRow - 1, StartColumn + 2).AddToNotAllowedSymbols(CurrentSymbol)
                            self.__GetCell(StartRow - 2, StartColumn + 2).AddToNotAllowedSymbols(CurrentSymbol)
                            self.__GetCell(StartRow - 2, StartColumn + 1).AddToNotAllowedSymbols(CurrentSymbol)
                            self.__GetCell(StartRow - 2, StartColumn).AddToNotAllowedSymbols(CurrentSymbol)
                            self.__GetCell(StartRow - 1, StartColumn).AddToNotAllowedSymbols(CurrentSymbol)
                            self.__GetCell(StartRow - 1, StartColumn + 1).AddToNotAllowedSymbols(CurrentSymbol)
                            if (WildCardInPattern):
                                return 15
                            else:
                                return 10
                    #END CHANGE
                except:
                    pass
        return 0
```

Suitable variable to count the number of wild cards available:

```
class Puzzle():
    def __init__(self, *args):
        if len(args) == 1:
            #CHANGE
            self.__WildCardsLeft = 2
            #END CHANGE
            self.__Score = 0
            self.__SymbolsLeft = 0
            self.__GridSize = 0
            self.__Grid = []
            self.__AllowedPatterns = []
            self.__AllowedSymbols = []
            self.__LoadPuzzle(args[0])
        else:
            #CHANGE
            self.__WildCardsLeft = 2
            #END CHANGE
```

Modification to the `AttemptPuzzle` method:

```
    def AttemptPuzzle(self):
        Finished = False
        while not Finished:
            self.DisplayPuzzle()
            print("Current score: " + str(self.__Score))
            #CHANGE
            print("Wild cards will allow two symbols to be placed on top of each other and still be matched for eac
            print("You currently have " + str(self.__WildCardsLeft) + " wild cards left")
            #CHANGE
            Row = -1
            Valid = False
            while not Valid:
```

"

```
            CurrentCell = self.__GetCell(Row, Column)
            if CurrentCell.CheckSymbolAllowed(Symbol):
                #CHANGE
                if (not CurrentCell.IsEmpty() and CurrentCell.GetSymbol() != "@"):
                    if (self.__WildCardsLeft > 0):
                        self.__Grid[(self.__GridSize - Row) * self.__GridSize + Column - 1] = WildCardCell()
                        self.__WildCardsLeft -=1
                    else:
                        print("You have used both your wildcards for this puzzle.")
                        continue
                else:
                    CurrentCell.ChangeSymbolInCell(Symbol)
                #END CHANGE
                AmountToAddToScore = self.CheckforMatchWithPattern(Row, Column)
                if AmountToAddToScore > 0:
```


Testing

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

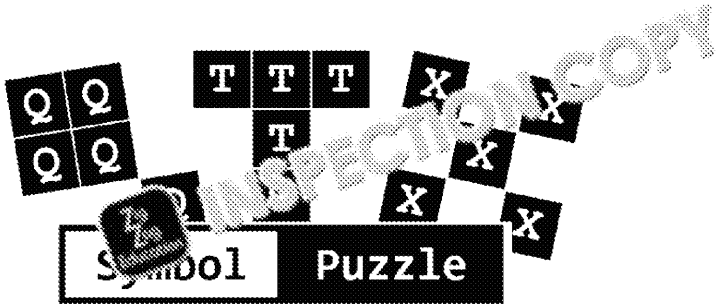
Preview of Answers Ends Here

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	