**A black and white logo

Description automatically generated**

**Theory Questions (Mark Scheme)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Question** | | **Answer** | **Mark** | **Guidance** |
| 1 | a | The selection statement calls the Puzzle constructor with just one parameter, the filename, if it has been entered (greater than 0 length) [1], which will initialise the Puzzle using the data from the specified text file [1].  However, if the filename is not supplied (length is 0) then a Puzzle size of 8 is specified [1] with a second parameter (max pieces) set to 60% of the grid spaces available (8\*8\*0.6) // 38 [1] and the blocks will be randomly generated [1]. | 4 | MAX 4 |
| 1 | b | Overloading | 1 |  |
| 2 | a | 1 mark each for any of the following:  Constants have meaningful identifiers which make code easier to follow/understand/update/maintain  Constants only have to be updated in one place if the value changes | 2 | MAX 2 |
| 2 | b | OOP classes and objects often model reality [1] making it easier to understand and create [1]  OOP classes encapsulate behaviour and attributes [1] by hiding the implementation [1] // by including properties and methods inside the interface  OOP allows many programmers to work together [1] as classes can be developed and tested separately [1] | 4 | MAX two points with up to 1 mark for each development |
| 3 | a | A string of the symbols on the grid is generated using the same helix pattern as the pattern match string for each pattern [1] and they are then compared by calling the MatchPattern method. If all the non \* symbols match then MatchPattern will return True indicating a successful match [1]. | 2 |  |
| 3 | b | When the string from the grid is compared to the pattern match, if one of the non \* symbols doesn’t match then it is unsuccessful | 1 |  |
| 3 | c | When a pattern has been matched, each Cell in the 3 x 3 grid is updated to have the current symbol added to its SymbolsNotAllowed list. | 1 |  |
| 4 |  | QQ..Q..QQ | 1 |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Question** | | **Answer** | **Mark** | **Guidance** |
| 5 | a | Inherits means that the child class gains all of the (non-private) methods/behaviours/functions/subroutines of the parent [1] and all of the attributes/variables/characteristics of the parent [1] | 2 |  |
| 5 | b | By using inheritance, a BlockedCell can be treated as a Cell [1] through polymorphism [1].  BlockedCell (is truly a specialisation of Cell and) declutters the Cell class [1] by keeping all of the features of BlockedCell contained/encapsulated [1]. We can override CheckSymbolAllowed [1] and make sure that it always returns False for a BlockedCell [1]. | 4 | MAX 4 from 2+2 (advantage and development twice) |
| 5 | c | Override is where you have a method in a child class with the same name as a method in the parent class [1] but a different implementation [1] | 2 |  |
| 6 | a | The iterative statement keeps looping until an integer is entered [1].  It does this by setting the loop condition (Valid) = false to start and only to true if an integer has been currently entered [1].  The try…except means that if a non-integer is entered then a Runtime exception will be raised [1] and the trail of execution will be interrupted so that valid will not be set to true and the except clause will run instead [1]. | 4 |  |
| 6 | b | Raising an exception will cause the current thread of execution to stop [1] and will terminate the program with a Runtime Error [1] unless the exception is handled [1] | 2 | MAX 2 |
| 7 | a | Overriding the CheckSymbolAllowed method | 1 |  |
| 7 | b | The ability for an object to be treated as one of its parents [1] allowing it to be grouped with other objects that have a common parent [1] but for methods that are called to actually be resolved on the real class of the object rather than the method in the parent [1] | 3 |  |
| 8 | a | GetCell has an interface that requires a row and a column [1] which suggests that the data structure will be two-dimensional [1] when in fact the implementation uses a one-dimensional list/array [1] | 3 |  |
| 8 | b | The 57th element [1] of Grid will be returned [1] (which is equivalent to row 1, col 2) | 2 |  |
| 9 | a | GetSymbol calls the method IsEmpty() to check if the Cell is empty [1] and then returns the (hard-coded) symbol ‘-’ if it is [1]  IsEmpty returns true if the length of the symbol is 0, otherwise false [1] | 3 |  |
| 9 | b | A private attribute can only be used with the current class [1] whereas a protected attribute can also be used in any classes that inherit from the current class // child class [1] | 2 |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 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 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]   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 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Question** | | **Answer** | **Mark** | **Guidance** |
| 15 |  | 1 mark for each of the following:   Creating a function with the correct name and returning an integer value   Creating a way of checking the symbols required (such as a variable) and the number of matches found   Having a for loop that goes through all possible locations for the pattern   AND only goes through the possible locations for the pattern   Checking if the pattern is found accurately and returning 50 when it is   Returning 0 when the pattern is not found  // Assuming indexing starts at 1  FUNCTION CheckForSpecialPattern : INTEGER  BEGIN  STRING specialPattern = “QQQQTTXXTXXTTTXX”  FOR row = 8 DOWNTO 4  FOR col = 1 TO 5  STRING patternString = “”  patternString = patternString + GetCell(row,col).GetSymbol()  patternString = patternString + GetCell(row,col+1).GetSymbol()  patternString = patternString + GetCell(row,col+2).GetSymbol()  patternString = patternString + GetCell(row,col+3).GetSymbol()  patternString = patternString + GetCell(row-1,col).GetSymbol()  patternString = patternString + GetCell(row-1,col+1).GetSymbol()  patternString = patternString + GetCell(row-1,col+2).GetSymbol()  patternString = patternString + GetCell(row-2,col+3).GetSymbol()  patternString = patternString + GetCell(row-2,col).GetSymbol()  patternString = patternString + GetCell(row-2,col+1).GetSymbol()  patternString = patternString + GetCell(row-2,col+2).GetSymbol()  patternString = patternString + GetCell(row-2,col+3).GetSymbol()  patternString = patternString + GetCell(row-3,col).GetSymbol()  patternString = patternString + GetCell(row-3,col+1).GetSymbol()  patternString = patternString + GetCell(row-3,col+2).GetSymbol()  patternString = patternString + GetCell(row-3,col+3).GetSymbol()  IF patternString == specialPattern  THEN return 50  NEXT col  NEXT row  RETURN 0  END FUNCTION | 6 |  |