****

**Theory Questions (Mark Scheme)**

| **Question** | | **Suggested Solution** | **Mark** | **Marking Guidance** |
| --- | --- | --- | --- | --- |
| **1** | (a) | The Choice variable is converted to lower case to ensure the input is case-insensitive. This way, both “Y” and “y” will be treated the same, simplifying the logic. [1] | 1 mark | **A:** any reasonable suggestion |
| (b) | The TrainingGame variable determines whether the game is in training mode or random mode. In training mode, predefined numbers and targets are used, while in random mode, the numbers and targets are randomly generated. [1] | 1 mark | **A:** any reasonable suggestion |
| **2** |  | The repeated call to DisplayState() updates the game display, showing the current targets, available numbers, and score after every move. [1] This helps the player track their progress and remaining resources. [1] | 2 marks | 1 mark for each point (MAX. 2) |
| **3** | (a) | UserInputInRPN holds the user's input infix expression converted into Reverse Polish Notation (RPN) format, which is used for easier evaluation. This is a list of string “symbols” used in the expression. [1] | 1 mark | **R:** only mentioning that it is a list |
| (b) | The function converts the user's input to RPN and iterates through the expression/list. [1] For each valid number used in the expression/list that matches a number in NumbersAllowed, the number is removed from the list to prevent reuse. [1] | 2 marks | **A:** any similar explanation |
| **4** | (a) | The player's score increases when their evaluated expression matches one of the current targets. [1] | 1 mark |  |
| (b) | Setting the target to -1 marks it as "hit" or "used". The aim of the game is to turn the targets into -1 so that the player can survive for longer as the game loop ends when a non -1 target is encountered in position 0 of the Targets list. [2] | 2 marks | **A:** any similar explanation |
| **5** | (a) | The pattern string ^[0-9]+$ means that the Item string that was input must: [1]  a. Start with  b. one or more numbers in the range 0–9  c. End after this digit / these digits  The function will check whether the string matches these criteria so therefore the Item string is a valid integer and can be cast to the integer data type for further processing. [1] | 2 marks |  |
| (b) | Without the + character, the function would only match single-digit integers, i.e. 13 would be invalid. [1] | 1 mark | **A:** any other example |
| **6** | (a) | In RPN, operators follow their operands, allowing for expressions to be evaluated without parentheses. Operands are pushed onto a stack, and when an operator is encountered, the appropriate number of operands is popped from the stack for evaluation. The result is then pushed back onto the stack. [2] | 2 marks |  |
| (b) | The function would crash. No specific exception handling is present for such cases. [1] | 1 mark |  |
| **7** |  | In training mode, NumbersAllowed is populated with a fixed list of numbers [2, 3, 2, 8, 512]. In random mode, the list is filled by generating random numbers between 1 and MaxNumber inclusive until it has five elements. [2] | 2 marks |  |
| **8** | (a) | A stack is used to ensure operators are applied in the correct order, based on their precedence. Higher-precedence operators are applied before lower-precedence ones, and the stack helps manage this ordering. [2] | 2 marks |  |
| (b) | After the while loop has completed, which will have removed any/all operators (currently only \* or /) of a higher precedence from the stack, a selection statement is used to handle matching precedence operators. [1]  A matching precedence operator will be popped off the top of the stack onto the output prior to the new operator being pushed back onto the stack. [1] | 2 marks |  |
| **9** | (a) | GetTarget() generates a random target number between 1 and MaxTarget to be added to the Targets list. [1] | 1 mark |  |
| (b) | In the training game, the Targets list is initialised as a list of literal integers. [1] In the random game, the CreateTargets() function is used to build a list containing five -1s representing empty cells followed by 15 random integers in the range 1 to 50 inclusive. [1] | 2 marks |  |
| **10** | (a) | [1] for each correct label | 6 marks |  |
| (b) | * CheckIfUserInputValid() validates the expression using a regular expression string and will  return true. [1] * ConvertToRPN() converts the expression into a list of RPN symbol strings. [1] * CheckNumbersUsedAreAllInNumbersAllowed() ensures that all numbers used in the expression are available in the NumbersAllowed list. They are, so it will return true. [1] * CheckIfUserInputEvaluationIsATarget() evaluates the expression using EvaluateRPN(), compares it to the targets, and adjusts the score by +2 if it matches a target. It also changes the target to a -1  in order to flag that target as “solved”. It is, so this function will return true and the new score of  the player. [1] * RemoveNumbersUsed() removes the numbers used in the expression from the available numbers and overwrites the NumbersAllowed list variable. [1] * FillNumbers() refills the available numbers with either a fresh copy of the training game numbers  [2, 3, 2, 8, 512] or random numbers between 1 and MaxNumber inclusive (10 default). It also overwrites the NumbersAllowed list variable. [1]   [6] – 1 mark for each subroutine call and associated description. | 6 marks | **A:** any similar explanation |
| **11** | (a) | Exception handling can be useful to catch and manage runtime errors, such as invalid input or calculation errors (e.g. division by zero). It ensures that the program doesn’t crash and can recover gracefully by informing the user of the issue. [1] | 1 mark |  |
| (b) | Exception handling could be added in EvaluateRPN() to catch division by zero errors, allowing the program to display an error message and request a new input without crashing. [1] | 1 mark |  |
| **12** | (a) | The GameOver variable is set to true when the first target in the Targets list is no longer available (i.e. Targets[0] != -1). [1] | 1 mark |  |
| (b) | It prevents the loop from running indefinitely, ensuring that the game ends when all relevant gameplay conditions have been met. [1] | 1 mark |  |
| **13** |  | Any 2 from:   * The highest score could be stored in a file or a database. [1] * At the start of the game, the file/database would be read to retrieve the previous high score. [1] * After each game, if the new score exceeds the old high score, the file/database would be updated with the new value. [1] | 2 marks | **A:** any 2 points from 3 |
| **14** | (a) | CreateTargets / FillNumbers / ConvertToRPN / RemoveNumberUsed / UpdateTargets [1] | 1 mark |  |
|  | (b) | TrainingGame [1] | 1 mark |  |
|  | (c) | UserInput, Number [1] | 1 mark |  |
|  | (d) | RemoveAt / Add [1] | 1 mark |  |
|  | (e) | MaxTarget / MaxNumber / MaxNumberOfTargets [1] | 1 mark |  |
| **15** |  | Any 2 from:   * + - means 1 or more of preceding character/sequence [1] * [0-9]+ means 1 or more digits from 0 to 9 [1] * ([0-9]+[\\+\\-\\\*\\/])+ means 1 or more sequences of a number (operand) followed by  an operator [1] | 2 marks | **A:** any 2 points from 3 |
| **16** |  | Because regular expressions do not support recursion. [1]  A regular expression cannot track the opening and closing of brackets / a regular expression has no notion of “state”. [1] | 1 mark | **A:** 1 from 2 points given |
| **17** |  | The precedence of the current operator is compared to the precedence of the operator on the top of the Operators stack. [1]  While it is greater, the top of the stack is repeatedly popped onto UserInputInRPN output. [1]  A final single check is carried out to compare whether the top of the stack has the same precedence as the current operator. If it has, the stack is popped once more onto the UserInputInRPN output. [1] | 3 marks |  |
| **18** |  | Decomposition: The program is broken into smaller tasks, each handled by specific functions. [1]  Abstraction: Demonstrated by hiding the complexity of certain tasks behind clear, high-level functions that perform specific roles. [1] | 2 marks |  |
| **19** |  | O(n). For n elements in the target list, n operations will be carried out. [1] | 1 mark |  |