TARGET CLEAR

A LEVEL AQA

PRE-RELEASE 2025

Theory Questions

These questions are designed to test your understanding of the skeleton code. Many of these are similar to the kinds of question you can expect to see in Section C of the Paper 1 exam. However, sub-questions that are more than 2 marks are rarely seen in this section – these more involved questions are here to challenge your understanding of the code.

These questions refer to the **Preliminary Material** and the **Skeleton Program**, but **do not** require any additional programming

TOTAL MARKS: 57

1. This question is about the **Main()** subroutine.

(a) Explain why the **Choice** variable is converted to lower case in the **Main()** subroutine. [1]

(b) Explain the purpose of the **TrainingGame** variable in the program. [1]

2. This question is about the **PlayGame()** subroutine. It repeatedly calls **DisplayState()**.

Explain the purpose of this repeated call and how it contributes to the gameplay. [2]

3. This question is about the **RemoveNumbersUsed()** function.

(a) Identify what **UserInputInRPN** represents within this function. [1]

(b) Explain the logic used to remove numbers from the **NumbersAllowed** list. [2]

4. This question is about the function **CheckIfUserInputEvaluationIsATarget()** and how it works to modify the player's score.

(a) What condition needs to be met to increase the player's score? [1]

(b) Why is the target set to -1 after it has been evaluated successfully? [2]

5. This question is about the function **CheckValidNumber()**. The function uses a regular expression.

(a) Explain the purpose of using the regular expression in this function and how this regular expression works to validate user input. [2]

(b) What could happen if the regular expression pattern was changed to ^**[0-9]$** by removing the **+** character? [1]

6. This question is about the **EvaluateRPN()** function. It evaluates expressions in Reverse Polish Notation (RPN).

(a) Briefly describe how Reverse Polish Notation works and how it is evaluated using a stack. [2]

(b) What would happen if an invalid operation (e.g. division by zero) is attempted in this function? [1]

7. Examine the function **FillNumbers()**. It works differently in training and random game modes.

Explain how the list **NumbersAllowed** is populated in training mode versus random mode. [2]

8. This question is about the function **ConvertToRPN()**. Operators are stored in a stack while operands are processed immediately.

(a) Explain why a stack is used to manage operators in this function. [2]

(b) How does the function handle operators of equal precedence? [2]

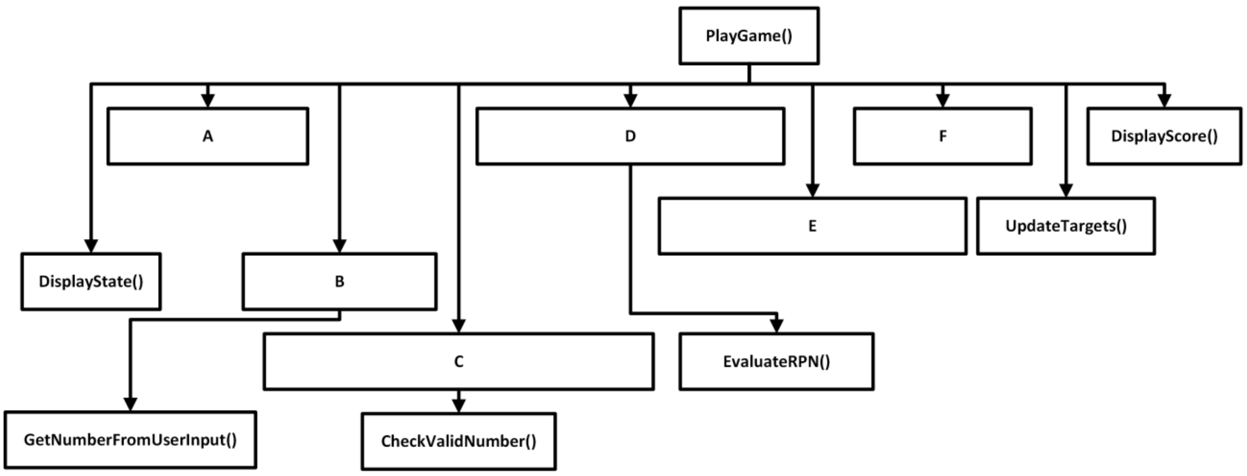
9. This question is about the function **CreateTargets()**.

(a) What is the role of the **GetTarget()** function within **CreateTargets()**? [1]

(b) Explain how the **Targets** list is initialised differently at the start of the game. [2]

10. This question is about the **PlayGame()** subroutine.

(a) Below is a hierarchy chart for **PlayGame()**. Name the six user-defined subroutines labelled A to F. [6]



(b) Describe the purpose/functionality of each of the six labelled subroutines from part (a). As part of your description, you can assume that the player enters a **valid expression** that uses **only available numbers** and will **correctly hit** one of the targets. [6]

11. This question refers to the use of exception handling in programming.

(a) Why might it be useful to use exception handling in a program like this, especially for user input? [1]

(b) Provide an example of where exception handling could be implemented in this program to improve robustness. [1]

12. The question is about the **PlayGame()** subroutine. The subroutine contains a loop that continues until the **GameOver** variable is true.

(a) Explain the criteria for setting the **GameOver** condition to be True. [1]

(b) Why is it important to have a condition like **GameOver** to end a loop? [1]

13. Imagine you want to add a feature to permanently store the highest score achieved in the game.

Explain where you would store this information and how you would retrieve it when needed. [2]

14. State an identifier for / name of:

(a) A user-defined function that returns a list [1]

(b) A Boolean variable within the **Main()** subroutine [1]

(c) A string variable within the function **GetNumberFromUserInput()** [1]

(d) A list method that is used within the function **UpdateTargets()** [1]

(e) An integer variable within the function **Main()** [1]

15. This question is about the **CheckIfUserInputValid()** function. Inside it there is a regular expression.

"^([0-9]+[\\+\\-*\\/])+[0-9]+$"

How does the regular expression make use of the + meta-character? [2]

16. Explain why a regular expression could not be adapted to check the validity of a mathematical expression with (indefinitely nested) brackets but BNF syntax could be used. [1]

17. This question is about the **ConvertToRPN()** function. Explain how the function makes use of the **Precedence** dictionary. [3]

18. Explain how this program demonstrates the concepts of abstraction and decomposition through the use of functions. [2]

19. This question is about the **UpdateTargets()** function. The function implements a *shunting* of the targets down by one position each time it is called. What is the time complexity for this operation? [1]

END OF QUESTIONS