Question 1:

A: The EQUALS(a, b) algorithm helps to determine if two words, a and b, are identical. To do this, I would compare each character of the words, ensuring they match at every corresponding position. If the lengths of the words are not the same, I would immediately say that they are not identical and return false. If the lengths match, I will proced to compare each character.

If at any point I find a mismatch between a character in word a and the corresponding character in word b, I would return false. On the other hand, if all characters in both words match, I would return true, indicating that the words are indeed identical. For example, when checking EQUALS(x, y), I would find that "computer" and "algorithm" have different characters at the first position, so I would return flse. But when comparing EQUALS(x, z), I would check that all characters in "computer" match the corresponding characters in "computer," leading me to return true.

B: FUNCTION wordExisted(book, word):

FORs i = 1 TO length(book) DO

IF EQUALS(book[i], word) THEN

RETURN tre

END IF

END FOR loops

RTURN false

END FUNCTION

C: O(n)

D: FUNCTION MATCHED(text, pattern, s):

if s + length(pattern) > length(text) THEN

RETURN false

END IF

FOR j = 1 TO length(pattern) DO

IF text[s + j] != pattern[j] THEN

RETURN flse

END IFs

END FOR loops

RTURN true

END FUNCTION

E: FUNCTION TEXTMATCHING(text, pattern):

FOR s = 1 TO length(text) - length(pattern) + 1 DO

IF MATCH(text, pattern, s) THEN

RETURN true

END IF

END FOR loops

RTURN false

END FUNCTION

D: FUNCTION ReverseOrder(a, b):

mergedList = [] // Initialize an empty list to store the merged result

i = length(a) - 1 // Start from the last element of list a

j = length(b) - 1 // Start from the last element of list b

WHILE i >= 0 AND j >= 0 DO

IF a[i] > b[j] THEN

mergedList.append(a[i]) // Append the larger element from list a

i = i - 1 // Move to the previous element in list a

ELSE

mrgedList.append(b[j]) // Append the larger element from list b

j = j - 1 // Move to the previous element in list b

END IF

END WHLE Loop

// Add remining elements from list a, if any

WHILE i >= 0 DO

mergedList.append(a[i])

i = i - 1

END WHILE loop

// Add remaining elements from list b, if any

WHILE j >= 0 DO

mergedList.append(b[j])

j = j - 1

END WHLE Loop

RTURN mergedList // Returns the merged result in decreasing order

END FUNCTION

Q3:

A: The output would be <a, "Hello", "World!"> or

["Hello", "World!"]

B: FUNCTION REVERSE(s):

newStack = new STACK // Create a new stack to store the reversed elements

WHILE s is not empty DO

element = s.pop() // Pop the top element from the original stack

newStack.push(element) // Push the element into the new stack

END WHILE loop

RETURNs newStack // Returns the new stack with reversed elements

END FUNCTION

C: FUNCTION TOSTACK(q):

newStack = new STACK // Create a new stack to store the elements

WHILE q is not empty DO

element = q.dequeue() // Dequeue the element from the head of the queue

newStack.push(element) // Push the element into new stack

END WHILE loop

RETURNs newStack // Return the new stack with elements from queues

END FUNCTION

D)

1: The final output would be [35]

2: (3 + 4) \* 5 is performed, The numbers 3 and 4 are pushed onto the stack T then added together once the + operator is found, afterwards 5 is added to the stack then once \* is found it becomes 4\*5 which is 35.

3: The numbers 4 and 5 would be pushed into stack followed by an \* encounter which would multiply 4 and 5. The number 3 would be pushed into the stack followed by the + operator. This would result in 3 + (4\*5).

4:

The CALCULATE function is a method of performing mathematical operations. It uses two main data structures are a stack labelled s and another one labelled t.

it checks the top card of the s pile. If it finds a number, it moves this card to the t plate. However, if it finds a mathematical operator, it removes the top two number cards from the t plate, performs the required operation, and places the result back onto the s pile.

This cycle continues until the s pile is completely empty. At the end of the function, the final answer of the calculation is the sum that remains in the s pile.

To sm it up, the CALCULATE function is a clever method of manipulating values between two piles (s and t), while performing simple arithmetic operations to eventually arrive at the final result.