# Question 1

Four common operations in data structures are:

1. Access: Retrieving an element from the data structure.
2. Insertion: Adding an element to the data structure.
3. Deletion: Removing an element from the data structure.
4. Traversal: Visiting each element of the data structure in a systematic way.

# Question 2

Advantages of Linked List over Array:

1. Dynamic Size: Linked lists are dynamic data structures that can grow or shrink on demand. In contrast, arrays have a fixed size that needs to be declared at the time of initialization.
2. Efficient insertion and deletion: In a linked list, inserting or deleting an element from the middle of the list is a simple and fast operation. On the other hand, in an array, inserting or deleting an element from the middle of the array requires shifting all the elements to the right or left, respectively.

Disadvantages of Linked List over Array:

1. Random access is not allowed: In an array, elements can be accessed in constant time using their index. However, in a linked list, elements can only be accessed sequentially from the beginning of the list.
2. Extra memory is required: Linked lists require extra memory for storing the pointers to the next node. In contrast, arrays do not require any extra memory for storage apart from the elements themselves.

# Question 3

Big O notation is a mathematical notation used to describe the complexity of an algorithm. It is commonly used in computer science to classify algorithms according to their runtime or space complexity. Algorithms with a lower O notation are generally more efficient than those with a higher notation.

# Question 4

The sentence "The algorithm’s running time is O (n²)" means that the algorithm's time complexity is quadratic. This means that the amount of time the algorithm takes to execute increases exponentially with the size of the input data. In other words, as the input data grows larger, the time required to run the algorithm increases much faster than the size of the input itself.

# Question 5

To answer your question, the time complexity of the given function is O(n), where n is the length of the input array. This is because the function iterates over the array twice, one time to calculate the sum of its elements and another time to calculate the product of its elements. Although the function has two loops, they run separately one after the other, rather than nested within each other. Therefore, the time complexity is proportional to n, making it linear.