**Approach**

The program follows these steps:

1. **Initialization**:
   * total\_waiting\_time: A variable to keep track of the total waiting time for all customers.
   * current\_time: A variable to keep track of the current time as we process each customer.
2. **Iterate through each customer**:
   * For each customer, extract the arrival time and service time.
   * If the current\_time is less than the customer's arrival time, update current\_time to the arrival time. This ensures that the server starts serving the customer as soon as they arrive if the server is idle.
   * Calculate the waiting\_time for the customer as the difference between current\_time and arrival plus the service time.
   * Add the waiting\_time to total\_waiting\_time.
   * Update current\_time by adding the service time to it, indicating that the server is busy for the duration of the service time.
3. **Calculate and return the average waiting time**:
   * Divide total\_waiting\_time by the number of customers to get the average waiting time.

**Time Complexity**

The time complexity of this program is determined by the loop that iterates through each customer.

* **Initialization**: Initializing variables total\_waiting\_time and current\_time takes constant time, (O(1)).
* **Iteration through customers**: The loop iterates through each customer exactly once. If there are (n) customers, this loop runs (n) times.
  + Inside the loop, all operations (accessing array elements, arithmetic operations, and updating variables) take constant time, (O(1)).

Therefore, the time complexity of the entire program is (O(n)), where (n) is the number of customers.

**Space Complexity**

The space complexity is determined by the amount of extra space used by the program, excluding the input data.

* The program uses a few extra variables (total\_waiting\_time, current\_time, arrival, service, and waiting\_time), all of which take constant space, (O(1)).

Thus, the space complexity of the program is (O(1)).