General ideas:

         Since the customers will be served first-come first-served, declare a linked list LL of customer-Nodes, where the linked list will mimic a queue. Every node has several data fields:customerID,  arrivalTime, queueLength, and waitingTime. The queueLength is the number of people waiting in front of this customer at the time of his/her arrival.

         Int T = read that value from the top of the customers file;

         Read the customers file one customer at a time, inserting each customer Node to the end of LL. Only customerID and arrivalTime of each node are set;

         Declare another linked list, finalLL, of fully processed nodes;

         This problem is really a **simulation** of a real-time real-life situation. Therefore, you should have a master clock initialized to 9am, and mimicking the real-time clock. However, instead of having the clock move by individual seconds, you should update the clock value only at important events (this approach is referred to as *event-driven simulation*). Specifically, whenever a person is done being served, this clock will be incremented by T seconds if there is another person actually waiting in line, or will be forwarded up to the arrival time of the next customer in the LL list if no one is “actually waiting” in line.

         Therefore, the core of the program should be a while-loop iterating as long as the clock is < 5pm and the list LL is not empty. In each iteration, the head of the queue LL is the next customer to be served; that customer may have been waiting for a while, or will “arrive” after the current time of the clock. Either way, that customer should be dequeued from LL, and its waitingTime should be computed (and set) from: (1) the new clock value after it is changed as explained in the previous bullet, and (2) the arrival time of that customer. Also compute and set the queueLength of this customer node. Observe that queueLength can be derived from waitingTime and the value of T. Now, the customer node should be added tofinalLL.

         You should declare the following variables: maxBreakTime, totalBreakTime, numOfCustomersServed, and maxQueueLength; initialize them to 0. They represent at any given moment the maximum break time so far, the total idle time so far, the number of customers served so far, and the maximum number of people simultaneously waiting in line as observed so far, respectively. In each iteration of the while loop, those variables should be updated.

         At the end of the while loop, you need to check if the clock is still before 5pm. If so, the remaining time until 5pm is one last break. Therefore, add this break time to thetotalBreakTime, and update the maxBreakTime if necessary.

         Note that to be able to compare times and measure time difference (in seconds), you may want to write two functions. The 1st function takes two “actual” time recordings as input, and returns whether the first time is >=  the second time or < than the second time. The 2nd function takes two “actual” time recordings as input, and returns the difference between them in seconds.

         Now, you have LL empty, finalLL complete, and the variables maxBreakTime, totalBreakTime, numOfCustomersServed, and maxQueueLength have their final values computed. You can answer any type of query from one of those four variables or from the nodes in finalLL.