Simulation of an Appliance Store

Pickup System

1. Project Idea

The project simulated an in-store pickup system in an appliance store. When customers order products online, they can choose the pickup method. For customers who do not require express delivery services, in-store pickup is their best choice. We have a list of customer appointments, which includes the customer's name, and the name of the item ordered. However, the customer's arrival time is random, and the time to wait for the goods to be picked is randomly generated within a fixed time limit. Therefore, we use a program to generate a customer pickup list sorted by arrival time, including the customer and the product name ordered, the time the customer arrived at the store, the time the customer waited for the clerk to select the product, and the time the customer checked, inspected, and signed for the goods at the cashier desk, and used this time information to generate the customer's stay in the store.

It uses a **priority queue** to store customer data sorted by arrival time, ensuring that customers are processed in the order they arrive. The processed customers are stored in a vector, which facilitates writing the data to a file. The core data storage structure is a struct, which holds customer details such as arrival time, waiting time, and total time in-store. This approach efficiently reads, processes, and outputs customer pickup data

2. Data Structures Used

* **Structure(**struct Customer{}**):**
  + Used to store the basic information of a customer, including their name, ordered product, arrival time, waiting time, and checkout time.
  + The member functions **getArrivalTimeInMinutes()** and **getTotalTimeInStore()** are used to calculate the customer's arrival time in minutes and their total time spent in the store.
* **Priority Queue** **(**priority\_queue<Customer, vector<Customer>, CompareCustomerArrival>**):**
  + Purpose: Sorts customers by their arrival time, ensuring that the earliest arriving customer is processed first.
  + The **CompareCustomerArrival** struct defines the comparison rule, making the priority queue prioritize the customer with the smallest arrival time.
* **Vector** (vector<Customer> completedCustomers):
  + Purpose: Stores the list of customers who have completed their pickup process, including their arrival time, waiting time, and total time spent in the store.
  + This allows for easy traversal and facilitates writing the processed customer data to an output file.

3. Interface Components Used

* **File I/O**:
  + The program reads customer data(name) from an input file (customer.txt), processes the data, and writes the results to an output file (completed\_customers.txt).
  + ifstream is used to read from the input file, while ofstream is used for writing the output file.
* **Standard Input/Output**:
  + cout is used for displaying messages, such as reading the customer data, reporting errors, and completing tasks. This provides feedback to the user during execution. Since **using namespace std;** is included in the code, the **std::** prefix is not required for standard library components.

5.Program Code

The full source code is submitted along with this documentation. The code ensures that:

* Customers are served in order of arrival using a priority queue.
* The package pickup process is realistic, with randomized package search times and checkout times.
* Data is correctly read from and written to files for accurate simulation.

6. Conclusion

This program effectively simulates the package pickup process at an express delivery point. It uses a **priority queue** for managing customer arrivals and a **vector** for recording completed customers. The program also demonstrates efficient use of **file I/O** to read and write structured data. By incorporating random arrival, search, and checkout times, this simulation realistically models real-world package pickup scenarios and can be extended for further studies or testing.