Data Structures and Algorithms

Project: Restaurant Management System

1. Overview

This project simulates the operation of a **Restaurant Management System** using data structures and algorithms to efficiently manage order flow, chef assignment, and timing-based events.
The objective is to design and implement an event-driven simulation that models how orders are received, assigned to chefs, prepared, and completed — while tracking waiting times, service times, and overall performance.

2. Simulation Concept

The simulation proceeds through **discrete timesteps**, each representing one unit of time.
At every timestep, one or more of the following events may occur:

- New orders arrive.
- Orders are assigned to chefs.
- Orders are completed.
- Orders are canceled or promoted to VIP.
- Chefs take or finish their breaks.

After processing all actions for the current timestep, the counter increments and the next timestep begins.

3. System Entities

3.1 Orders

Each order is defined by the following information:

- **Request Timestep (RT):** Time when the order request was received.
- **Order Type:**
  - **VIP** — High-priority orders; always served first.
  - **Vegan** — Orders prepared by vegan chefs only.
  - **Normal** — Standard orders.
- **Order Size:** Number of dishes in the order.
- **Order Price:** Total money to be paid by the customer.

3.2 Chefs

Each chef has the following attributes:

- **Chef Type:**
  - **VIP Chef** — Highly skilled; can prepare any order type, but primarily VIP.
  - **Vegan Chef** — Specialized in plant-based food; serves only vegan orders.
  - **Normal Chef** — Standard chef for normal orders.
- **Speed:** Number of dishes the chef can prepare per timestep.
- **Break Duration:** Rest period after preparing a certain number of orders.
- **Orders Before Break:** Number of consecutive orders before a break.

All data about chefs is loaded from the input file.

4. Order Assignment Rules

Orders are assigned to chefs according to the following rules:

1. **VIP Orders:**
   - Served first.
   - Prefer **VIP Chefs**, then **Normal**, then **Vegan** (if available).
   - Within VIP orders, the selection follows a **priority equation** based on multiple factors (see below).
2. **Vegan Orders:**

    - Served only by **Vegan Chefs**.

3. **Normal Orders:**
    - Served by **Normal Chefs** first; if unavailable, assign to **VIP Chefs**.
    - Can be **canceled** or **promoted** to VIP manually or automatically.

If an order cannot be assigned during a timestep, it remains waiting for the next.

5. Priority Equation for VIP Orders

VIP orders must be stored in a **priority queue**.
The system should use a **weighted priority equation** that reflects both urgency and profitability.
A reasonable equation might be:

```
Priority = (Money × α) / (OrderSize × β × (CurrentTime − RT + 1))
```

Where:

- α and β are weighting factors chosen by the developer.
- Higher priority values indicate orders that should be served sooner.

6. Order Promotion and Cancellation Rules

- **Manual Promotion (P event):**
  A Normal order may be promoted to VIP by paying extra money (`ExtraMony` in input file).

- **Automatic Promotion:**
  If a Normal order waits longer than **AutoP** timesteps without assignment, it automatically becomes VIP.

- **Cancellation (X event):**
  A Normal order can be canceled as long as it is still waiting (not yet assigned).

All promotions and cancellations are event-driven and loaded from the input file.

7. Simulation Definitions

Term	Description
RT	Arrival (request) timestep.
GT	Assignment timestep (when chef starts cooking).
WT	Waiting Time = GT − RT
ST	Service Time (cooking duration).
FT	Finish Time = RT + WT + ST
In-Service Order	Currently being prepared by a chef.
Done Order	Finished order waiting for delivery or record.

8. Input and Output File Formats

8.1 Input File Format

```
N G V
SN SG SV
BO BN BG BV
AutoP
M
<event lines>
```

Where:

- **N, G, V:** Number of Normal, Vegan, and VIP chefs.
- **SN, SG, SV:** Speed for each chef type.
- **BO:** Number of orders before a chef takes a break.
- **BN, BG, BV:** Break durations for Normal, Vegan, and VIP chefs.
- **AutoP:** Number of timesteps before auto-promotion to VIP.

- **M:** Number of total events.

8.2 Event Line Formats

- **Arrival:**

  ```
  R TYP TS ID SIZE MONY
  ```

  Example:

  ```
  R N 7 1 15 110
  R V 9 3 21 300
  R G 12 4 53 42
  ```

- **Cancellation:**

  ```
  X TS ID
  ```

  Example:

  ```
  X 15 1
  ```

- **Promotion:**

  ```
  P TS ID ExtraMony
  ```

  Example:

  ```
  P 19 2 62
  ```

8.3 Output File Format

Each completed order must be written as:

```
FT ID RT WT ST
```

After all orders are processed, summary statistics are printed:

- Total number of orders and per-type counts.
- Total number of chefs and per-type counts.
- Average waiting time and service time.
- Percentage of automatically promoted orders.

9. Program Modes

Interactive Mode

Displays real-time simulation updates for each timestep.

Silent Mode

No screen output; only the final output file is produced.

10. Suggested Class Design

- **Restaurant** — Controls simulation and manages data lists.
- **Order** — Represents order information and state.
- **Chef** — Represents chef details and workload.
- **Event (abstract)** — Base class for all event types.
    - **ArrivalEvent** — Creates a new order.
    - **CancelEvent** — Cancels a Normal order.
    - **PromoteEvent** — Promotes a Normal order to VIP.
- **UI** — Displays simulation status (interactive mode only).

11. Recommended Data Structures

Entity	Recommended DS	Description
Events	Queue	Processed in chronological order.
VIP Orders	Priority Queue	Sorted by calculated priority.
Vegan Orders	Queue	FCFS.
Normal Orders	Queue	FCFS with cancellation & promotion support.
Chefs	Queues	Separate queues for available, in-service, and break chefs.
Done Orders	List	Stores completed orders for output.

12. Implementation Guidelines

- Use incremental simulation (step-by-step).
- Pass references or pointers; avoid global variables.
- Move objects between lists using pointers, not copies.
- Ensure output is sorted by **Finish Time (FT)**.
- Stop simulation when all events are processed and all orders are done.

13. Bonus and Extensions (Optional Ideas)

- Different chefs of the same type with variable speeds or break durations.
- Simulate random injuries reducing chef speed temporarily.
- Introduce additional order or chef types (e.g., dessert or delivery specialists).

14. Sample Scenario

At timestep 25:

- A **Normal order** arrives and is assigned to a Normal chef with speed 2.
- A **Vegan order** arrives but all Vegan chefs are busy → queued.
- At timestep 27, a Vegan chef becomes available → picks up the order.
- At timestep 30, a Normal order is automatically promoted to VIP after waiting longer than AutoP.

15. Learning Outcomes

Through this project, students will:

- Design **event-driven simulations**.
- Apply **queues, stacks, and priority queues** effectively.
- Manage **dynamic state transitions** in an operational system.
- Implement **object-oriented design** with inheritance and encapsulation.
- Handle **structured file input/output**.
- Analyze **performance metrics** and scheduling efficiency.
- Build simulation logic applicable to real-time systems.