**Project Report: Ice-Cream Factory Simulation**

**Group members**

**Roll number: 22k-4351**

**Roll number: 22k-4598**

**Roll number: 22k-4559**

**1. Introduction**

The project is a simulation of an "Ice-Cream Factory Problem" designed to demonstrate concurrency and resource management using semaphores in a multithreaded environment. It also integrates graphical representation with the SFML (Simple and Fast Multimedia Library) for visual interaction.

**2. Objective**

To simulate an ice-cream shop with multiple customers, each undergoing a sequence of processes: acquiring a ticket, selecting a flavor, adding toppings, and completing payment. The goal is to ensure synchronized access to limited resources while visually displaying customer progress.

**3. Features**

1. **Concurrency Management:**
   * Multiple customers are simulated as threads.
   * Semaphores are used to synchronize shared resources like tickets, flavors, toppings, and payment counters.
2. **Graphical User Interface (GUI):**
   * The SFML library is used for rendering a user-friendly interface.
   * Buttons allow the user to specify the number of customers (1–5).
   * Animated progress of customers is shown in the GUI, from entering the shop to payment completion.
3. **Revenue Calculation:**
   * Each customer contributes a fixed amount of revenue ($100).
   * The total revenue is calculated and displayed.

**4. Implementation**

**a. Languages and Libraries**

* **C++**: Core programming language.
* **SFML**: Used for graphics rendering.
* **Pthreads**: Manages threading.
* **POSIX Semaphores**: Ensures resource synchronization.

**b. Core Components**

1. **Main Process Workflow:**
   * Users select the number of customers via GUI buttons.
   * Threads are created for each customer.
   * Each thread executes the ice\_cream() function, simulating customer actions.
2. **Semaphore Control:**
   * Semaphores (ticket, flavour, topping, payment) regulate access to critical resources.
   * Prevents race conditions and ensures fairness.
3. **Animation and Display:**
   * Customers are visually represented as sprites.
   * Progress through stages is displayed via animations and text on the GUI.

**c. Class Structure**

* Button: Handles GUI button interactions.
* draw(): Renders customer progress and updates the GUI dynamically.

**5. Execution Flow**

1. Start the application.
2. Use the GUI to select the number of customers.
3. Watch customer progress through:
   * Ticket acquisition.
   * Flavor selection.
   * Topping addition.
   * Payment and exit.
4. Total revenue is displayed at the end of the simulation.

**6. Challenges Addressed**

* **Synchronization**: Avoided deadlocks and race conditions using semaphores.
* **Concurrency**: Successfully managed multiple customer threads simultaneously.
* **User Interface**: Provided an engaging GUI to enhance user interaction.

**7. Applications**

* Educational tool for teaching concurrency and synchronization concepts.
* Basis for developing resource management systems in industries.

**8. Conclusion**

The Ice-Cream Factory simulation effectively demonstrates the use of multithreading, semaphores, and graphical programming to solve a synchronization problem. It offers a blend of technical rigor and visual appeal, making it suitable for both academic and demonstration purposes.

**Thank you!**