Project 3

(Duck Park)

CS 415 - Operating Systems

Fall 2025 - Prof. Jieyang Chen

Due date: **11:59 pm, Friday, Dec 5th, 2025**

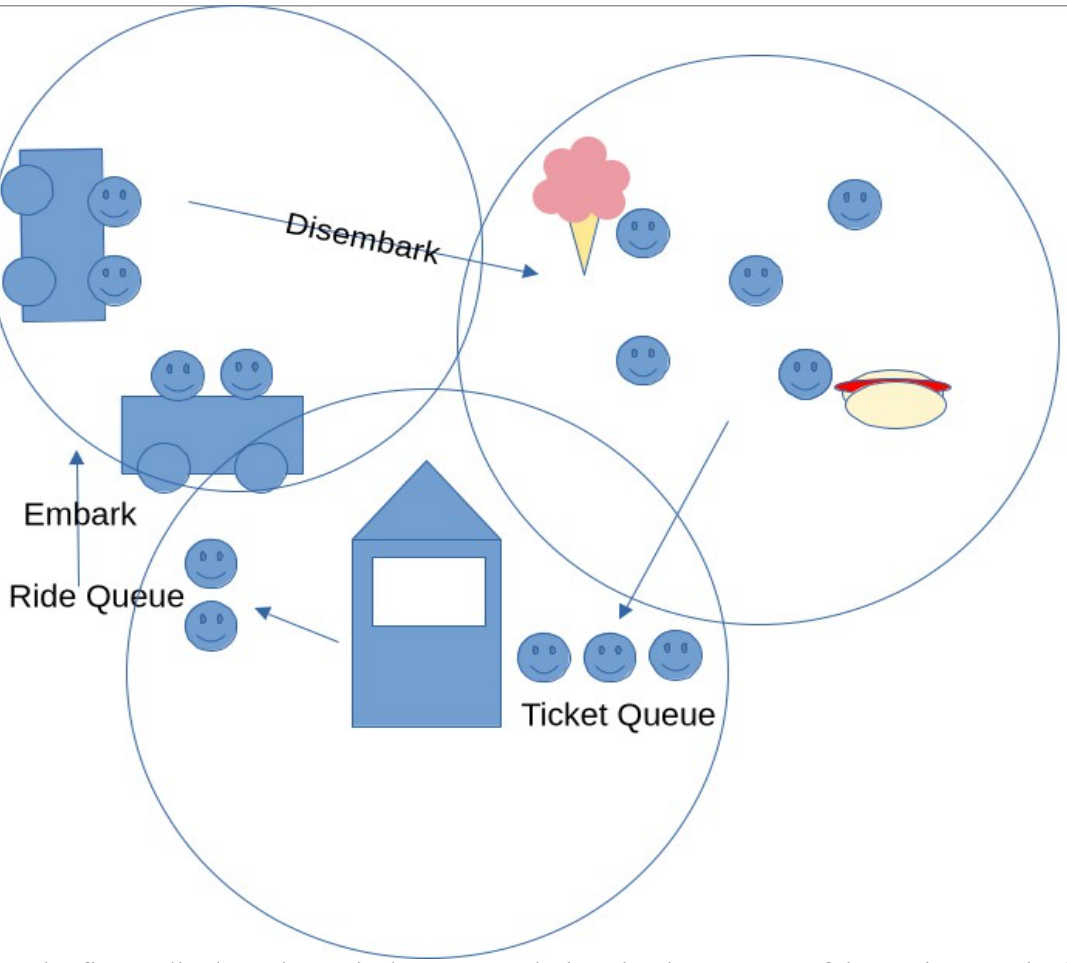
Introduction

In this project, we are going to use Pthread and synchronization tools to create a Parallel Discrete-Event Simulation (PDES) system to simulate a roller coaster amusement park called Duck Park.

- **Park:** The park opens when the simulation starts and closes after a give periods (e.g., after running for T seconds), and the simulation program should exit. Suppose there are N passenger threads and C car threads in the park.
- **Cars:** Each car is job is repeatedly loading passengers, running on the track, and unloading passengers.
  - Each car can hold up to P passengers, where P < N. Once the car is full it can go around the tracks immediately. When it is empty, it should wait indefinitely. When it is partially filled (has at least one passenger and less than P passengers), it can only departure if it has been waited for W seconds since the last passenger boarded.
  - Once a car departs, it should run on the track for R seconds.
  - Multiple cars can be on the track running concurrently, but only one car can be loading at a time. Since cars can't pass each other, they must unload in the same order they loaded. All passengers from one car must unload before any of the passengers from following cars.
  - Each car should have functions: load, run, and unload, and it should repeated invoke these functions in order.
- **Passengers:** Each passenger enters the park and exploration (let passenger threads sleep) for random time period (1-10s) before wanting to ride the coaster (maybe they are playing games, eating funnel cake, etc.).
  - Passengers need to get a ticket from a ticket booth. If multiple passengers want to buy, they have to wait in a Ticket Queue. Only one passenger can get a ticket at a time.
  - One they get tickets, each passenger need to enter a Ride Queue to wait for loading. You should place a bound (J) on the length of the Ride Queue. That is, if the queue is full (number of passengers in Ride Queue = J), a ticket cannot be issued. A passenger cannot leave any queue early by itself.
  - Once a passenger reach the front of the Ride Queue, it cannot board a car until the car has invoked load (notify the passenger that it can board).
  - After the ride, passengers cannot unboard until the car has invoked unload (notify passengers that they can unboard).
  - Each passenger should have functions: explore park, get ride ticket, enter ride queue, board car, and unboard car. As passenger repeatedly wants to explore the park and waits to take rides in a car, it should repeatedly invoke these functions in order.
- **Simulation inputs:** your simulation program should take the following input from user
  - N: number of passenger threads
  - C: number of car threads
  - P: capacity per car (maximum of passengers per car)
  - W: car waiting period
  - R: car ride duration

- T: park open duration
- J: Ride Queue max size

Illustration:



The figure displays three circles encapsulating the three states of the project: park, ticket, ride. In the first circle, there are five smiley faces representing threads that are hanging out in the park. From this group of smiley faces, and arrow points from them to the second conjoined circle where there is a queue of smiley faces at a ticket booth. This is a ticket buying queue. From the booth there is an arrow pointing to the third "ride" state that is labeled "embark" (load). Here, we see two full roller coasters on track with two smiley faces in each car. The ride queue represents the processes that are waiting to get in a car. There is a final arrow form the ride element back to the park labeled "disembark" (unload).

Deliverables

Part 1: Single-Threaded Solution

Develop and test each component with a single passenger thread and a single car thread to verify basic functionality. Terminal output should reflect the state of the system include actions make or status changes for each thread.

Part 2: Multi-threaded Solution

Test your implementation with multiple passenger threads entering and exiting from multiple car threads at varying times to ensure correct synchronization. You should choose to use any combination of **mutex lock**, **semaphore**, and **conditional variable** in the Pthread library to implement. Terminal output should reflect the state of the system include actions make or status changes for each thread.

Part 3: Monitor

Create a monitoring system using a difference process (e.g., child process) that:

- Reports on the state of all queues (ticket and ride)
- Regularly samples and displays system state
- Implements inter-process communication using pipes or mmap

See example outputs in the end of this document: Your output does **not** have to match the example here exactly, but it will be **verbose**, neat, and include the **time**. It should demonstrate that your simulation working seamlessly with parallelism and synchronicity.

Report

Write a 1-2 page report on your project using the sample report collection format given. Your report could also discuss your synchronization approach, potential deadlocks and how you avoided them. Explain design decisions and trade-offs made. Feel free to go over the limit if you wish. Report format and content suggestions are given in the report collection template. If you are not able to complete all 3 parts, state in your report which part you finished, so partial credit can be given.

Program Requirements

You must use the C programming language with the Pthread library for this project. No projects written in another programming language will be accepted. To achieve proper synchronization, you should choose to use any combination of **mutex lock**, **semaphore**, and **conditional variable** in the Pthread library. To achieve monitoring through a different process, you should use either POSIX shared memory or PIPE.

Here is a list of suggested functions (this is not a complete list, and not all functions need to be used)

- pthread_create()
- pthread_exit()
- pthread_join()
- pthread_mutex_lock()/pthread_mutex_unlock()
- sem_wait()/sem_timedwait()/sem_post()
- pthread_cond_wait()/pthread_cond_timedwait()/pthread_cond_signal()
- mmap() or pipe()

Source code: For a project to be accepted, the project must contain the following files and meet the following requirements

- **park.c**: this is the main program
- Additionally, you are allowed to add any other *.h and *.c files you wish. Make sure your code runs in the IX-DEV before submission.
- **Makefile**: Your project must include a standard make file. It must produce the executable with the following names: **park**
- **How to run**:
  USAGE:
  ./park [OPTIONS]

  OPTIONS:
  -n: number of passenger threads
  -c: number of car threads
  -p: capacity per car (maximum of passengers per car)
  -w: car waiting period
  -r: car ride duration
  -t: park open duration
  -j: Ride Queue max size
  -h: display help

  EXAMPLE:
  ./park -n 30 -c 4 -p 2 -w 3 -r 2 -t 60 -j 10

  You can optionally use the getopt() function in main to help you parse the command line option. You can also write your own parser without using getopt(). Note: regardless of how to decide to parse the input, you need to include **default options** in case some options are not provided by the users.
- **What to output**: Terminal output showing your simulation running in real time.

Submission Requirements

Once your project is done, do the following:

- Your executable should be able to run with the following command "./park [options]
- Open a terminal and navigate to the project folder. Compile your code in the IX-DEV with the -g, -pthread, and - lpthread flag.
- Run your code and take screenshots of the output as necessary (of each part).
- Create valgrind logs of each respective part:
  "valgrind --leak-check=full --tool=memcheck ./a.out > log*.txt 2>&1 "
- Tar/zip the project folder which should contain the following directories and content. Your project should have part1, part2 and part3 folders.
  1) part1
     - park.c
     - Any additional header file and their corresponding ".c" file
     - Makefile
     - Output (directory)
     - valgrind log

2) part2
    - park.c
    - Any additional header file and their corresponding ".c" file
    - Makefile
    - Output (directory)
    - valgrind log
3) part3
    - park.c
    - Any additional header file and their corresponding ".c" file
    - Makefile
    - Output (directory)
    - valgrind log

Valgrind can help you spot memory leaks in your code. As a general rule any time you allocate memory you must free it. Points will be deducted in the project for memory leaks so it is important that you learn how to use and read Valgrind's output. See (https:// valgrind.org/) for more details.

The naming convention of the zip/tar file while uploading to canvas Name_ProjectX (an example is given below)

- firstname: abie
- lastname: safdie
- Submission for: Project3
- So the name of the zip/tar file should be: abie_safdie_Project3.zip

Rubric

Part	Points	Description
1	30	Simulation runs correctly with a single passenger and a single car thread, All critical sections are well protected with locks, Correct usage of pthread_create, Correct usage of pthread_join. Park logic correct as described in the project description.
2	30	Simulation runs correctly with any number of passenger and car threads. Park logic correct as described in the project description.
3	20	Correct usage of Pipes for IPC. Accurately displays live states of queues.
Valgrind	10	No memory leak/errors. 1 point each until all 10 points are gone.
Report	10	1 - 2 page report

Checklist

Part 1: Single Passenger & Single Car (30 points)

- **Critical Sections Protection (10 points)**
    - Shared variable protection (2 points): All global variables (boarders, unboarders) use mutex/ semaphore
    - Queue operations synchronized (2 points): enqueue/dequeue operations are atomic
    - Ticket booth serialization (2 points): Only one passenger gets ticket at a time

- Car state protection (2 points): Car state changes are thread-safe
- No race conditions (2 points): Multiple runs produce consistent and logical output

- **Pthread Usage (8 points)**
  - Thread creation (2 points): pthread_create() successfully creates passenger and car threads
  - Thread joining (2 points): pthread_join() waits for all threads before termination
  - Function signatures (2 points): Thread functions use void* function(void* arg) format
  - Memory safety (2 points): Thread arguments allocated on heap, not stack
- **Synchronization Logic (12 points)**
  - Load constraint (2 points): Passengers cannot board until car invokes load()
  - Capacity constraint (4 points): Car waits for exactly C passengers before run()
  - Unload constraint (2 points): Passengers cannot unboard until car invokes unload()
  - Function ordering (2 points): Car follows load→run→unload sequence
  - Passenger lifecycle (2 points): Passenger completes park→ticket→board→unboard cycle

Part 2: Multiple Passengers & Cars (30 points)

- **Multi-threading Correctness (12 points)**
  - Concurrent passenger safety (3 points): Multiple passengers can safely get tickets/board
  - Deadlock prevention (3 points): No deadlocks occur with any valid input parameters
  - Thread completion (3 points): All N passengers complete exactly one ride
  - Resource cleanup (3 points): All threads terminate cleanly
- **Multi-car Constraints (10 points)**
  - Exclusive boarding (3 points): Only one car loads passengers at any time
  - Concurrent running (2 points): Multiple cars can be on track simultaneously
  - FIFO unloading (3 points): Cars unload in same order they loaded (verified by timestamps)
  - Complete carload rule (2 points): All passengers from car N unboard before car N+1 unboards
- **Advanced Features (8 points)**
  - Timeout mechanism (3 points): Cars depart after W seconds when partially filled
  - Queue bound enforcement (3 points): Tickets denied when boarding queue reaches capacity
  - Parameter compliance (2 points): Uses command-line parameters correctly

Part 3: IPC Monitoring (20 points)

- **IPC Implementation (8 points)**
  - Pipe/mmap creation (2 points): Successfully creates communication channel
  - Data transmission (3 points): Queue states transmitted between processes/threads
  - Resource management (3 points): Proper cleanup of file descriptors/memory
- **Queue State Display (12 points)**
  - Real-time updates (4 points): Queue contents update as passengers move through system
  - Accurate representation (4 points): Displayed queue matches actual system state
  - Formatted output (2 points): Queue contents clearly formatted with timestamps
  - Both queues shown (2 points): Both ticket queue and boarding queue displayed

Note

- 0/100 if your program does not compile.
- 10 points deduction if your makefile does not work
- 30 points deduction if pthread synchronization functions used but did not contribute to the

actual functionality

- Missing functionality caused by chain effects will not receive credit. (correctly implemented but does not work due to other mistakes)

Late Policy

- 10% penalty (1 day late)
- 20% penalty (2 days late)
- 30% penalty (3 days late)
- 100% penalty (>3 days late) (i.e. no points will be given to projects received after 3 days)

Example Output (Part 1)

- Number of passenger threads: 1
- Number of cars: 1
- Capacity per car: 1
- Car waiting period: 1
- Car ride duration: 1
- Park duration: 25 seconds
- Max ride queue size: 1

[Time: 0] Passenger 0 entered the park
[Time: 0] Passenger 0 is exploring the park
[Time: 0] Car 0 invoked load()
[Time: 4] Passenger 0 finished exploring, entering the ticket booth
[Time: 4] Passenger 0 entering the ticket queue
[Time: 6] Passenger 0 acquired a ticket
[Time: 6] Passenger 0 has entered the ride queue
[Time: 6] Passenger 0 is boarding
[Time: 6] Car 0 is full with 1 passengers
[Time: 6] Car 0 has departed to ride
[Time: 11] Car 0 has returned from the ride
[Time: 11] Car 0 has invoked unload()
[Time: 11] Passenger 0 unboarded
[Time: 11] Passenger 0 is exploring the park
[Time: 11] Car 0 invoked load()
[Time: 18] Passenger 0 finished exploring, entering the ticket booth
[Time: 18] Passenger 0 entering the ticket queue
[Time: 20] Passenger 0 acquired a ticket
[Time: 20] Passenger 0 has entered the ride queue
[Time: 20] Passenger 0 is boarding
[Time: 20] Car 0 is full with 1 passengers
[Time: 20] Car 0 has departed to ride
[Time: 25] Car 0 has returned from the ride
[Time: 25] Car 0 has invoked unload()
[Time: 25] Passenger 0 unboarded
[Time: 25] Passenger 0 is exploring the park
[Time: 25] Car 0 invoked load()

Example Output (Part 2)

- Number of passenger threads: 10
- Number of cars: 2
- Capacity per car: 2
- Car waiting period: 1
- Car ride duration: 1
- Park duration: 30 seconds
- Max ride queue size: 3

[Time: 0] Passenger 0 entered the park
[Time: 0] Passenger 0 is exploring the park
[Time: 0] Passenger 1 entered the park
[Time: 0] Passenger 1 is exploring the park

[Time: 0] Passenger 2 entered the park
[Time: 0] Passenger 2 is exploring the park
[Time: 0] Passenger 3 entered the park
[Time: 0] Passenger 3 is exploring the park
[Time: 0] Passenger 4 entered the park
[Time: 0] Passenger 4 is exploring the park
[Time: 0] Passenger 5 entered the park
[Time: 0] Passenger 5 is exploring the park
[Time: 0] Passenger 6 entered the park
[Time: 0] Passenger 6 is exploring the park
[Time: 0] Passenger 7 entered the park
[Time: 0] Passenger 7 is exploring the park
[Time: 0] Passenger 8 entered the park
[Time: 0] Passenger 8 is exploring the park
[Time: 0] Passenger 9 entered the park
[Time: 0] Passenger 9 is exploring the park
[Time: 0] Car 1 invoked load()
[Time: 2] Passenger 9 finished exploring, entering the ticket booth
[Time: 2] Passenger 9 entering the ticket queue
[Time: 3] Passenger 7 finished exploring, entering the ticket booth
[Time: 4] Passenger 0 finished exploring, entering the ticket booth
[Time: 4] Passenger 4 finished exploring, entering the ticket booth
[Time: 4] Passenger 9 acquired a ticket
[Time: 4] Passenger 9 has entered the ride queue
[Time: 4] Passenger 7 entering the ticket queue
[Time: 4] Passenger 9 is boarding
[Time: 5] Car 1 waiting period expired
[Time: 5] Car 1 has departed to ride
[Time: 5] Car 0 invoked load()
[Time: 6] Passenger 3 finished exploring, entering the ticket booth
[Time: 6] Passenger 5 finished exploring, entering the ticket booth
[Time: 6] Passenger 7 acquired a ticket
[Time: 6] Passenger 7 has entered the ride queue
[Time: 6] Passenger 0 entering the ticket queue
[Time: 6] Passenger 7 is boarding
[Time: 6] Car 1 has returned from the ride
[Time: 6] Car 1 has invoked unload()
[Time: 6] Passenger 9 unboarded
[Time: 6] Passenger 9 is exploring the park
[Time: 7] Passenger 1 finished exploring, entering the ticket booth
[Time: 7] Passenger 6 finished exploring, entering the ticket booth
[Time: 7] Car 0 waiting period expired
[Time: 7] Car 0 has departed to ride
[Time: 7] Car 1 invoked load()
[Time: 8] Passenger 2 finished exploring, entering the ticket booth
[Time: 8] Passenger 0 acquired a ticket
[Time: 8] Passenger 0 has entered the ride queue
[Time: 8] Passenger 4 entering the ticket queue
[Time: 8] Passenger 0 is boarding
[Time: 8] Car 0 has returned from the ride
[Time: 8] Car 0 has invoked unload()

[Time: 8] Passenger 7 unboarded
[Time: 8] Passenger 7 is exploring the park
[Time: 9] Passenger 9 finished exploring, entering the ticket booth
[Time: 9] Car 1 waiting period expired
[Time: 9] Car 1 has departed to ride
[Time: 9] Car 0 invoked load()
[Time: 10] Passenger 8 finished exploring, entering the ticket booth
[Time: 10] Passenger 4 acquired a ticket
[Time: 10] Passenger 4 has entered the ride queue
[Time: 10] Car 1 has returned from the ride
[Time: 10] Car 1 has invoked unload()
[Time: 10] Passenger 3 entering the ticket queue
[Time: 10] Passenger 4 is boarding
[Time: 10] Passenger 0 unboarded
[Time: 10] Passenger 0 is exploring the park
[Time: 11] Car 0 waiting period expired
[Time: 11] Passenger 0 finished exploring, entering the ticket booth
[Time: 11] Car 0 has departed to ride
[Time: 11] Car 1 invoked load()
[Time: 12] Passenger 3 acquired a ticket
[Time: 12] Passenger 3 has entered the ride queue
[Time: 12] Passenger 5 entering the ticket queue
[Time: 12] Passenger 3 is boarding
[Time: 12] Car 0 has returned from the ride
[Time: 12] Car 0 has invoked unload()
[Time: 12] Passenger 4 unboarded
[Time: 12] Passenger 4 is exploring the park
[Time: 13] Car 1 waiting period expired
[Time: 13] Car 1 has departed to ride
[Time: 13] Car 0 invoked load()
[Time: 14] Passenger 5 acquired a ticket
[Time: 14] Passenger 5 has entered the ride queue
[Time: 14] Passenger 1 entering the ticket queue
[Time: 14] Passenger 5 is boarding
[Time: 14] Car 1 has returned from the ride
[Time: 14] Car 1 has invoked unload()
[Time: 14] Passenger 3 unboarded
[Time: 14] Passenger 3 is exploring the park
[Time: 15] Car 0 waiting period expired
[Time: 15] Car 0 has departed to ride
[Time: 15] Car 1 invoked load()
[Time: 16] Passenger 7 finished exploring, entering the ticket booth
[Time: 16] Passenger 1 acquired a ticket
[Time: 16] Passenger 1 has entered the ride queue
[Time: 16] Passenger 6 entering the ticket queue
[Time: 16] Passenger 1 is boarding
[Time: 16] Car 0 has returned from the ride
[Time: 16] Car 0 has invoked unload()
[Time: 16] Passenger 5 unboarded
[Time: 16] Passenger 5 is exploring the park
[Time: 17] Car 1 waiting period expired

[Time: 17] Car 1 has departed to ride
[Time: 17] Car 0 invoked load()
[Time: 18] [Time: 18] Passenger 6 acquired a ticket
[Time: 18] Passenger 6 has entered the ride queue
Passenger 3 finished exploring, entering the ticket booth
[Time: 18] Passenger 2 entering the ticket queue
[Time: 18] Passenger 6 is boarding
[Time: 18] Car 1 has returned from the ride
[Time: 18] Car 1 has invoked unload()
[Time: 18] Passenger 1 unboarded
[Time: 18] Passenger 1 is exploring the park
[Time: 19] Car 0 waiting period expired
[Time: 19] Car 0 has departed to ride
[Time: 19] Passenger 1 finished exploring, entering the ticket booth
[Time: 19] Car 1 invoked load()
[Time: 20] Passenger 2 acquired a ticket
[Time: 20] Passenger 2 has entered the ride queue
[Time: 20] Passenger 9 entering the ticket queue
[Time: 20] Passenger 2 is boarding
[Time: 20] Car 0 has returned from the ride
[Time: 20] Car 0 has invoked unload()
[Time: 20] Passenger 6 unboarded
[Time: 20] Passenger 6 is exploring the park
[Time: 21] Car 1 waiting period expired
[Time: 21] Car 1 has departed to ride
[Time: 21] Car 0 invoked load()
[Time: 22] Passenger 4 finished exploring, entering the ticket booth
[Time: 22] Passenger 9 acquired a ticket
[Time: 22] Passenger 9 has entered the ride queue
[Time: 22] Passenger 8 entering the ticket queue
[Time: 22] Passenger 9 is boarding
[Time: 22] Car 1 has returned from the ride
[Time: 22] Car 1 has invoked unload()
[Time: 22] Passenger 2 unboarded
[Time: 22] Passenger 2 is exploring the park
[Time: 23] Passenger 5 finished exploring, entering the ticket booth
[Time: 23] Car 0 waiting period expired
[Time: 23] Car 0 has departed to ride
[Time: 23] Car 1 invoked load()
[Time: 24] Passenger 8 acquired a ticket
[Time: 24] Passenger 8 has entered the ride queue
[Time: 24] Passenger 0 entering the ticket queue
[Time: 24] Passenger 8 is boarding
[Time: 24] Car 0 has returned from the ride
[Time: 24] Car 0 has invoked unload()
[Time: 24] Passenger 9 unboarded
[Time: 24] Passenger 9 is exploring the park
[Time: 25] Passenger 2 finished exploring, entering the ticket booth
[Time: 25] Car 1 waiting period expired
[Time: 25] Car 1 has departed to ride
[Time: 25] Car 0 invoked load()

[Time: 26] Passenger 0 acquired a ticket
[Time: 26] Passenger 0 has entered the ride queue
[Time: 26] Passenger 7 entering the ticket queue
[Time: 26] Passenger 0 is boarding
[Time: 26] Car 1 has returned from the ride
[Time: 26] Car 1 has invoked unload()
[Time: 26] Passenger 8 unboarded
[Time: 26] Passenger 8 is exploring the park
[Time: 27] Passenger 6 finished exploring, entering the ticket booth
[Time: 27] Car 0 waiting period expired
[Time: 27] Car 0 has departed to ride
[Time: 27] Car 1 invoked load()
[Time: 28] Passenger 7 acquired a ticket
[Time: 28] Passenger 7 has entered the ride queue
[Time: 28] Passenger 3 entering the ticket queue
[Time: 28] Passenger 7 is boarding
[Time: 28] Passenger 8 finished exploring, entering the ticket booth
[Time: 28] Car 0 has returned from the ride
[Time: 28] Car 0 has invoked unload()
[Time: 28] Passenger 0 unboarded
[Time: 28] Passenger 0 is exploring the park
[Time: 29] Car 1 waiting period expired
[Time: 29] Car 1 has departed to ride
[Time: 29] Car 0 invoked load()

Example Output (Part 3)

- Number of passenger threads: 10
- Number of cars: 2
- Capacity per car: 2
- Car waiting period: 1
- Car ride duration: 1
- Park duration: 30 seconds
- Max ride queue size: 3

[Time: 0] Passenger 0 entered the park
[Time: 0] Passenger 0 is exploring the park
[Time: 0] Passenger 1 entered the park
[Time: 0] Passenger 1 is exploring the park
[Time: 0] Passenger 2 entered the park
[Time: 0] Passenger 2 is exploring the park
[Time: 0] Passenger 3 entered the park
[Time: 0] Passenger 3 is exploring the park
[Time: 0] Passenger 4 entered the park
[Time: 0] Passenger 4 is exploring the park
[Time: 0] Passenger 5 entered the park
[Time: 0] Passenger 5 is exploring the park
[Time: 0] Passenger 6 entered the park
[Time: 0] Passenger 6 is exploring the park
[Time: 0] Passenger 7 entered the park
[Time: 0] Passenger 7 is exploring the park

[Time: 0] Passenger 8 entered the park
[Time: 0] Passenger 8 is exploring the park
[Time: 0] Passenger 9 entered the park
[Time: 0] Passenger 9 is exploring the park
[Time: 0] Car 0 invoked load()
[Time: 2] Passenger 9 finished exploring, entering the ticket booth
[Time: 2] Passenger 9 entering the ticket queue
[Time: 3] Passenger 7 finished exploring, entering the ticket booth
[Time: 4] Passenger 0 finished exploring, entering the ticket booth
[Time: 4] Passenger 4 finished exploring, entering the ticket booth
[Time: 4] Passenger 9 acquired a ticket
[Time: 4] Passenger 9 has entered the ride queue
[Time: 4] Passenger 7 entering the ticket queue
[Time: 4] Passenger 9 is boarding

[Monitor] SYSTEM STATE =>
Ticket Queue: [Passenger 7, Passenger 0, Passenger 4,]
Ride Queue: []
Car status 0 LOADING (1/2 Passengers)
Car status 1 WAITING (0/2 Passengers)
Passengers in the park: 10 (6 exploring, 3 in queues, 1 waiting in a car, 0 riding)

[Time: 5] Car 0 waiting period expired
[Time: 5] Car 0 has departed to ride
[Time: 5] Car 1 invoked load()
[Time: 6] Passenger 3 finished exploring, entering the ticket booth
[Time: 6] Passenger 5 finished exploring, entering the ticket booth
[Time: 6] Passenger 7 acquired a ticket
[Time: 6] Passenger 7 has entered the ride queue
[Time: 6] Passenger 0 entering the ticket queue
[Time: 6] Car 0 has returned from the ride
[Time: 6] Car 0 has invoked unload()
[Time: 6] Passenger 7 is boarding
[Time: 6] Passenger 9 unboarded
[Time: 6] Passenger 9 is exploring the park
[Time: 7] Passenger 1 finished exploring, entering the ticket booth
[Time: 7] Passenger 6 finished exploring, entering the ticket booth
[Time: 7] Car 1 waiting period expired
[Time: 7] Car 1 has departed to ride
[Time: 7] Car 0 invoked load()
[Time: 8] Passenger 2 finished exploring, entering the ticket booth
[Time: 8] Passenger 0 acquired a ticket
[Time: 8] Passenger 0 has entered the ride queue
[Time: 8] Passenger 4 entering the ticket queue
[Time: 8] Passenger 0 is boarding
[Time: 8] Car 1 has returned from the ride
[Time: 8] Car 1 has invoked unload()
[Time: 8] Passenger 7 unboarded
[Time: 8] Passenger 7 is exploring the park
[Time: 9] Passenger 9 finished exploring, entering the ticket booth
[Time: 9] Car 0 waiting period expired

[Time: 9] Car 0 has departed to ride
[Time: 9] Car 1 invoked load()

[Monitor] SYSTEM STATE =>
Ticket Queue: [Passenger 4, Passenger 3, Passenger 5, Passenger 1, Passenger 6, Passenger 2, Passenger 9,]
Ride Queue: []
Car status 1 LOADING (0/2 Passengers)
Passengers in the park: 10 (2 exploring, 7 in queues, 0 waiting in a car, 1 riding)

[Time: 10] Passenger 8 finished exploring, entering the ticket booth
[Time: 10] Passenger 4 acquired a ticket
[Time: 10] Passenger 4 has entered the ride queue
[Time: 10] Passenger 3 entering the ticket queue
[Time: 10] Passenger 4 is boarding
[Time: 10] Car 0 has returned from the ride
[Time: 10] Car 0 has invoked unload()
[Time: 10] Passenger 0 unboarded
[Time: 10] Passenger 0 is exploring the park
[Time: 11] [Time: 11] Car 1 waiting period expired
[Time: 11] Car 1 has departed to ride
Passenger 0 finished exploring, entering the ticket booth
[Time: 11] Car 0 invoked load()
[Time: 12] Passenger 3 acquired a ticket
[Time: 12] Passenger 3 has entered the ride queue
[Time: 12] Passenger 5 entering the ticket queue
[Time: 12] Passenger 3 is boarding
[Time: 12] Car 1 has returned from the ride
[Time: 12] Car 1 has invoked unload()
[Time: 12] Passenger 4 unboarded
[Time: 12] Passenger 4 is exploring the park
[Time: 13] Car 0 waiting period expired
[Time: 13] Car 0 has departed to ride
[Time: 13] Car 1 invoked load()
[Time: 14] Passenger 5 acquired a ticket
[Time: 14] Passenger 5 has entered the ride queue
[Time: 14] Passenger 1 entering the ticket queue
[Time: 14] Passenger 5 is boarding
[Time: 14] Car 0 has returned from the ride
[Time: 14] Car 0 has invoked unload()
[Time: 14] Passenger 3 unboarded
[Time: 14] Passenger 3 is exploring the park

[Monitor] SYSTEM STATE =>
Ticket Queue: [Passenger 1, Passenger 6, Passenger 2, Passenger 9, Passenger 8, Passenger 0,]
Ride Queue: []
Car status 1 LOADING (1/2 Passengers)
Car status 0 WAITING (0/2 Passengers)
Passengers in the park: 10 (3 exploring, 6 in queues, 1 waiting in a car, 0 riding)

[Time: 15] Car 1 waiting period expired
[Time: 15] Car 1 has departed to ride
[Time: 15] Car 0 invoked load()
[Time: 16] Passenger 7 finished exploring, entering the ticket booth
[Time: 16] Passenger 1 acquired a ticket
[Time: 16] Passenger 1 has entered the ride queue
[Time: 16] Passenger 6 entering the ticket queue
[Time: 16] Passenger 1 is boarding
[Time: 16] Car 1 has returned from the ride
[Time: 16] Car 1 has invoked unload()
[Time: 16] Passenger 5 unboarded
[Time: 16] Passenger 5 is exploring the park
[Time: 17] Car 0 waiting period expired
[Time: 17] Car 0 has departed to ride
[Time: 17] Car 1 invoked load()
[Time: 18] Passenger 3 finished exploring, entering the ticket booth
[Time: 18] Passenger 6 acquired a ticket
[Time: 18] Passenger 6 has entered the ride queue
[Time: 18] Passenger 2 entering the ticket queue
[Time: 18] Passenger 6 is boarding
[Time: 18] Car 0 has returned from the ride
[Time: 18] Car 0 has invoked unload()
[Time: 18] Passenger 1 unboarded
[Time: 18] Passenger 1 is exploring the park
[Time: 19] Car 1 waiting period expired
[Time: 19] Car 1 has departed to ride
[Time: 19] Passenger 1 finished exploring, entering the ticket booth
[Time: 19] Car 0 invoked load()

[Monitor] SYSTEM STATE =>
Ticket Queue: [Passenger 2, Passenger 9, Passenger 8, Passenger 0, Passenger 7, Passenger 3, Passenger 1,]
Ride Queue: []
Car status 0 LOADING (0/2 Passengers)
Passengers in the park: 10 (2 exploring, 7 in queues, 0 waiting in a car, 1 riding)

[Time: 20] Passenger 2 acquired a ticket
[Time: 20] Passenger 2 has entered the ride queue
[Time: 20] Passenger 9 entering the ticket queue
[Time: 20] Passenger 2 is boarding
[Time: 20] Car 1 has returned from the ride
[Time: 20] Car 1 has invoked unload()
[Time: 20] Passenger 6 unboarded
[Time: 20] Passenger 6 is exploring the park
[Time: 21] Car 0 waiting period expired
[Time: 21] Car 0 has departed to ride
[Time: 21] Car 1 invoked load()
[Time: 22] Passenger 4 finished exploring, entering the ticket booth
[Time: 22] Passenger 9 acquired a ticket
[Time: 22] Passenger 9 has entered the ride queue
[Time: 22] Passenger 8 entering the ticket queue

[Time: 22] Passenger 9 is boarding
[Time: 22] Car 0 has returned from the ride
[Time: 22] Car 0 has invoked unload()
[Time: 22] Passenger 2 unboarded
[Time: 22] Passenger 2 is exploring the park
[Time: 23] Passenger 5 finished exploring, entering the ticket booth
[Time: 23] Car 1 waiting period expired
[Time: 23] Car 1 has departed to ride
[Time: 23] Car 0 invoked load()
[Time: 24] Passenger 8 acquired a ticket
[Time: 24] Passenger 8 has entered the ride queue
[Time: 24] Passenger 0 entering the ticket queue
[Time: 24] Passenger 8 is boarding
[Time: 24] Car 1 has returned from the ride
[Time: 24] Car 1 has invoked unload()
[Time: 24] Passenger 9 unboarded
[Time: 24] Passenger 9 is exploring the park

[Monitor] SYSTEM STATE =>
Ticket Queue: [Passenger 0, Passenger 7, Passenger 3, Passenger 1, Passenger 4, Passenger 5,]
Ride Queue: []
Car status 0 LOADING (1/2 Passengers)
Car status 1 WAITING (0/2 Passengers)
Passengers in the park: 10 (3 exploring, 6 in queues, 1 waiting in a car, 0 riding)

[Time: 25] Passenger 2 finished exploring, entering the ticket booth
[Time: 25] Car 0 waiting period expired
[Time: 25] Car 0 has departed to ride
[Time: 25] Car 1 invoked load()
[Time: 26] Passenger 0 acquired a ticket
[Time: 26] Passenger 0 has entered the ride queue
[Time: 26] Car 0 has returned from the ride
[Time: 26] Car 0 has invoked unload()
[Time: 26] Passenger 7 entering the ticket queue
[Time: 26] Passenger 0 is boarding
[Time: 26] Passenger 8 unboarded
[Time: 26] Passenger 8 is exploring the park
[Time: 27] Passenger 6 finished exploring, entering the ticket booth
[Time: 27] Car 1 waiting period expired
[Time: 27] Car 1 has departed to ride
[Time: 27] Car 0 invoked load()
[Time: 28] Passenger 7 acquired a ticket
[Time: 28] Passenger 7 has entered the ride queue
[Time: 28] Passenger 3 entering the ticket queue
[Time: 28] Passenger 8 finished exploring, entering the ticket booth
[Time: 28] Passenger 7 is boarding
[Time: 28] Car 1 has returned from the ride
[Time: 28] Car 1 has invoked unload()
[Time: 28] Passenger 0 unboarded
[Time: 28] Passenger 0 is exploring the park

[Time: 29] Car 0 waiting period expired
[Time: 29] Car 0 has departed to ride
[Time: 29] Car 1 invoked load()

========== PARK CLOSED ==========

[Monitor] FINAL STATISTICS:
Total Simulation time: [Time: 30]
Total Passengers Served: 12
Total Rides: 12
Average Ticket Queue Seconds: 8.8
Average Ride Queue Seconds: 0.0
Average Car Utilization: 50.0 Percent (1.0/2 Passengers Per Ride)