CMA6134 Computational Methods Coding Assignment

Title: Car Arrival Simulator for a Car Wash Center

LECTURER : DR TONG HAU LEE
LECTURE SESSION: TC1L
TUTORIAL SESSION: TT1L

NAME	**ID**
VENGGADANAATHAN A/L K. SALVAM	1231303562
PRAVEEN KUMAR A/L PASKARAN	1191202561

Introduction

The purpose of this report is to develop and analyze a car arrival simulator for a car wash center with three wash bays. The simulator models a queuing system where cars arrive, queue up in a single lane, and proceed to an available wash bay. Each car owner may opt for a different car wash service, with service types randomly generated for each owner. This simulation aims to estimate various performance metrics of the car wash center, such as average waiting time, average service time, and the probability of cars having to wait in the queue.

Simulation Details

Problem Description

The car wash center has three wash bays, each with different service times. Cars arrive randomly and form a single queue. When a wash bay becomes available, the next car in line proceeds to that bay. The service time required by each car and the inter-arrival time between cars are generated randomly. The objective is to simulate this process and gather data to evaluate the performance of the car wash center.

Objectives

The main objectives of the simulation are:

1. To model the arrival and service process at a car wash center.
2. To generate random service times and inter-arrival times using different random number generators.
3. To simulate the queuing process and record the times for arrival, service, and departure for each car.
4. To evaluate the performance metrics such as average waiting time, average service time, and probability of waiting in the queue.

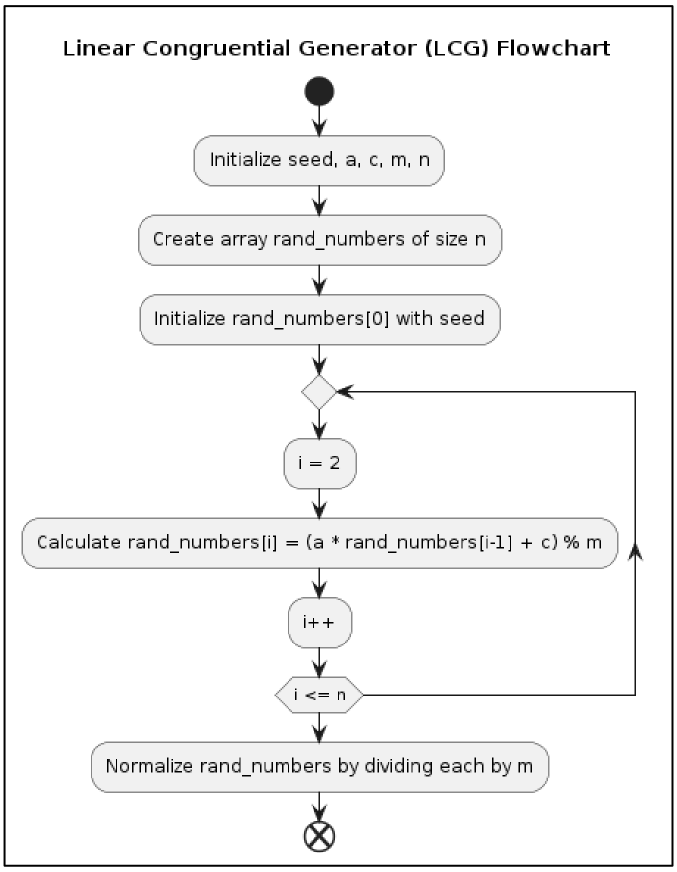
Assumptions

The following assumptions are made in this simulation:

1. Cars arrive at the car wash center in a single lane.
2. There are three wash bays, each with a different distribution of service times.
3. The inter-arrival times and service times are generated using random number generators.
4. Each car requires one of the available service types, chosen randomly.
5. The simulation runs until a user-defined number of cars have been processed.

FLOWCHARTS

1) Rand_generator.m file


Linear Congruential Generator (LCG) Flowchart
Initialize seed, a, c, m, n
Create array rand_numbers of size n
Initialize rand_numbers[0] with seed
i = 2
Calculate rand_numbers[i] = (a * rand_numbers[i-1] + c) % m
i++
i <= n
Normalize rand_numbers by dividing each by m

- **Start**: The process begins with initializing the necessary parameters and setting up the environment for generating random numbers.

- **Initialize Parameters**: The initial parameters for the LCG are set, which include the seed (initial value), multiplier (`a`), increment (`c`), modulus (`m`), and the number of random numbers to generate (`n`).

- **Create Array**: An array named `rand_numbers` of size `n` is created to store the sequence of random numbers.

- **Initialize First Element**: The first element of the `rand_numbers` array is initialized with the seed value provided.

- **Loop to Generate Random Numbers**:

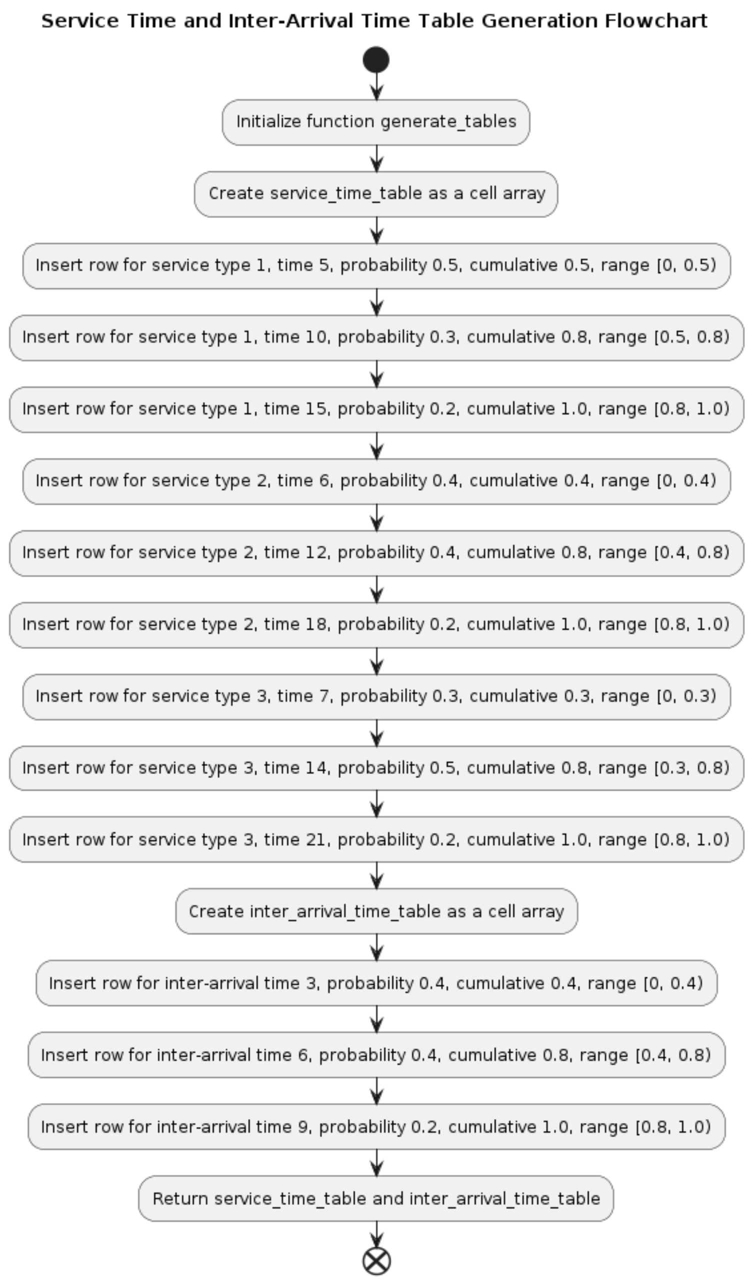
    - The loop starts with the index `i` set to 2, as the first element is already initialized.
    - In each iteration, the next random number is calculated using the formula `(a * rand_numbers[i-1] + c) % m` and stored in the `rand_numbers` array at the current index `i`.
    - The index `i` is then incremented by 1.

- **Repeat Condition**: The loop continues to iterate, generating and storing random numbers, until the index `i` exceeds `n`, ensuring that exactly `n` random numbers are generated.

- **Normalize Random Numbers**: After all random numbers are generated, they are normalized to be within the range of 0 to 1 by dividing each element in the `rand_numbers` array by the modulus `m`.

- **End**: The process concludes, resulting in an array `rand_numbers` filled with normalized random numbers, ready for further use or analysis.

2) generate_tables.m file


Service Time and Inter-Arrival Time Table Generation Flowchart
Initialize function generate_tables
Create service_time_table as a cell array
Insert row for service type 1, time 5, probability 0.5, cumulative 0.5, range [0, 0.5)
Insert row for service type 1, time 10, probability 0.3, cumulative 0.8, range [0.5, 0.8)
Insert row for service type 1, time 15, probability 0.2, cumulative 1.0, range [0.8, 1.0)
Insert row for service type 2, time 6, probability 0.4, cumulative 0.4, range [0, 0.4)
Insert row for service type 2, time 12, probability 0.4, cumulative 0.8, range [0.4, 0.8)
Insert row for service type 2, time 18, probability 0.2, cumulative 1.0, range [0.8, 1.0)
Insert row for service type 3, time 7, probability 0.3, cumulative 0.3, range [0, 0.3)
Insert row for service type 3, time 14, probability 0.5, cumulative 0.8, range [0.3, 0.8)
Insert row for service type 3, time 21, probability 0.2, cumulative 1.0, range [0.8, 1.0)
Create inter_arrival_time_table as a cell array
Insert row for inter-arrival time 3, probability 0.4, cumulative 0.4, range [0, 0.4)
Insert row for inter-arrival time 6, probability 0.4, cumulative 0.8, range [0.4, 0.8)
Insert row for inter-arrival time 9, probability 0.2, cumulative 1.0, range [0.8, 1.0)
Return service_time_table and inter_arrival_time_table

- **Start**: The process begins with the function `generate_tables` being called.

- **Create Service Time Table**: The function creates a cell array named `service_time_table` to store the details of service times for different service types.

- **Insert Rows for Service Time Table**

- **Create Inter-arrival Time Table**: The function creates a cell array named `inter_arrival_time_table` to store the details of inter-arrival times.

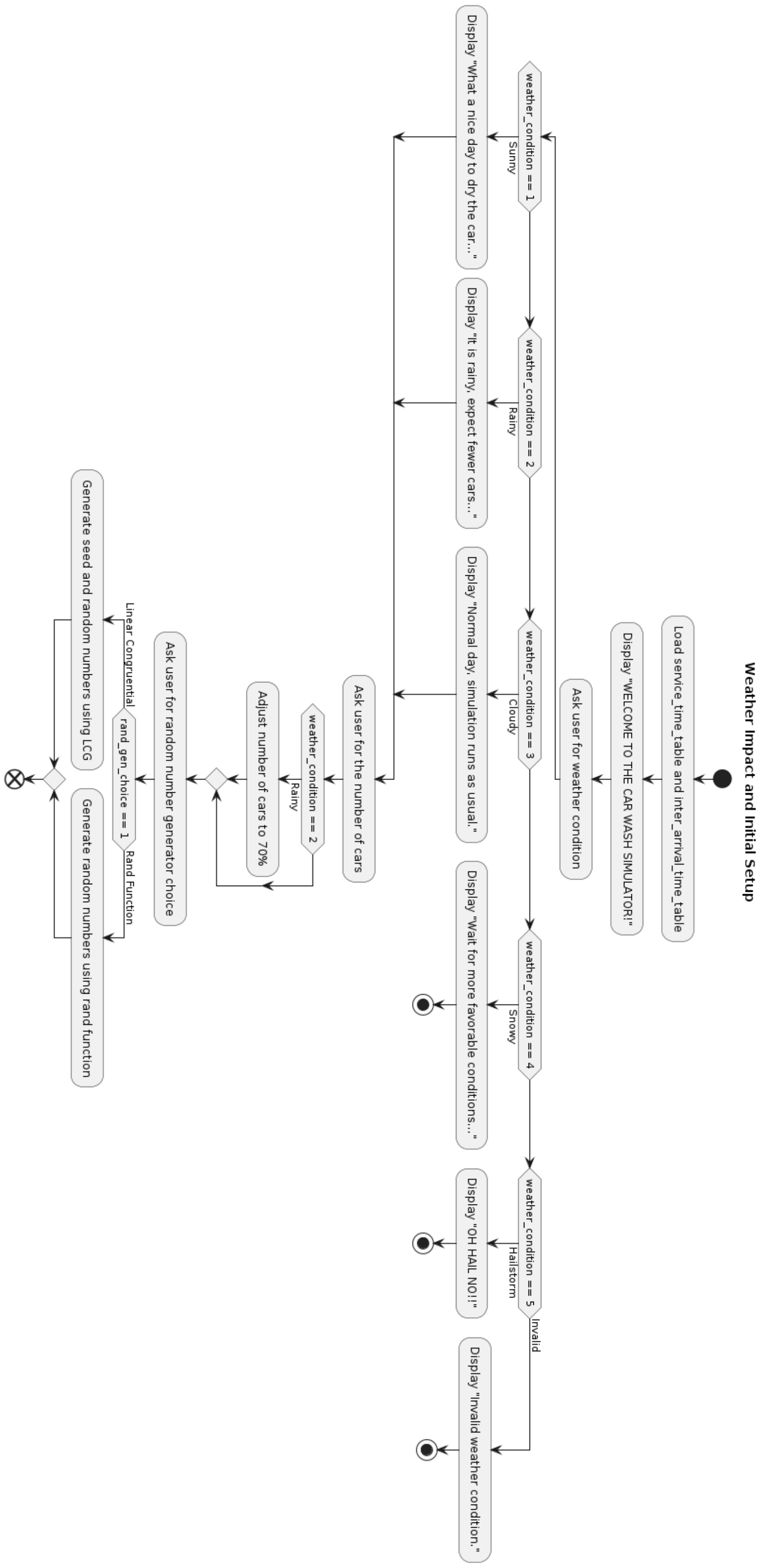
- **Insert Rows for Inter-arrival Time Table**:
  - The first row is inserted for an inter-arrival time of 3, with a probability of 0.4, a CDF value of 0.4, and a range of [0, 0.4).
  - The second row is inserted for an inter-arrival time of 6, with a probability of 0.4, a CDF value of 0.8, and a range of [0.4, 0.8).
  - The third row is inserted for an inter-arrival time of 9, with a probability of 0.2, a CDF value of 1.0, and a range of [0.8, 1.0).

- **Return Tables**: After populating both tables, the function returns the `service_time_table` and `inter_arrival_time_table`.

- **End**: The process ends, and the function exits.

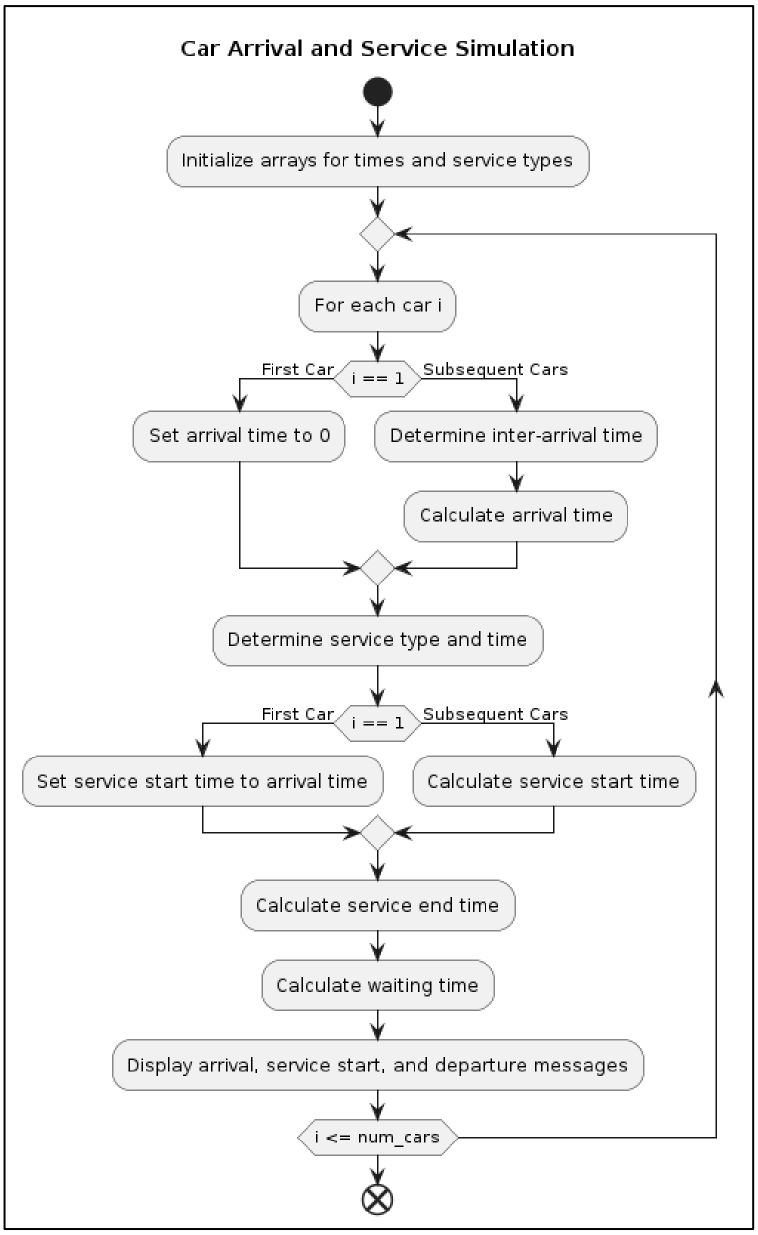
3) car_wash_simulator (main file)

- Weather Impact and initial setup


Weather Impact and Initial Setup
Load service_time_table and inter_arrival_time_table
Display "WELCOME TO THE CAR WASH SIMULATOR!"
Ask user for weather condition
weather_condition == 1
Sunny
Display "What a nice day to dry the car..."
weather_condition == 2
Rainy
Display "It is rainy, expect fewer cars..."
weather_condition == 3
Cloudy
Display "Normal day, simulation runs as usual."
weather_condition == 4
Snowy
Display "Wait for more favorable conditions..."
weather_condition == 5
Hailstorm
Display "OH HAIL NO!!"
Invalid
Display "Invalid weather condition."
Ask user for the number of cars
weather_condition == 2
Rainy
Adjust number of cars to 70%
Ask user for random number generator choice
rand_gen_choice == 1
Linear Congruential
Generate seed and random numbers using LCG
Rand Function
Generate random numbers using rand function

- **Start**: The process begins with loading the service time and inter-arrival time tables.
- **Display Welcome Message**: Display "WELCOME TO THE CAR WASH SIMULATOR!" to the user.
- **Ask for Weather Condition**: Prompt the user to input the current weather condition by entering a corresponding number (1 for Sunny, 2 for Rainy, 3 for Cloudy, 4 for Snowy, 5 for Hailstorm).
- **Check Weather Condition**:
    - If the weather is Sunny, display "What a nice day to dry the car..."
    - If the weather is Rainy, display "It is rainy, expect fewer cars..."
    - If the weather is Cloudy, display "Normal day, simulation runs as usual."
    - If the weather is Snowy, display "Wait for more favorable conditions..." and exit the script.
    - If the weather is Hailstorm, display "OH HAIL NO!!" and exit the script.
    - If the input is invalid, display "Invalid weather condition." and exit the script.
- **Ask for Number of Cars**: Prompt the user to input the number of cars.
- **Adjust Number of Cars for Rainy Weather**: If the weather is Rainy, adjust the number of cars to 70% of the input value.
- **Ask for Random Number Generator Choice**: Prompt the user to choose the random number generator (1 for Linear Congruential, 2 for Rand function).
- **Validate and Generate Random Numbers**:
    - If the user chooses Linear Congruential, generate a seed and use the LCG to generate random numbers.
    - If the user chooses Rand function, use the Rand function to generate random numbers.

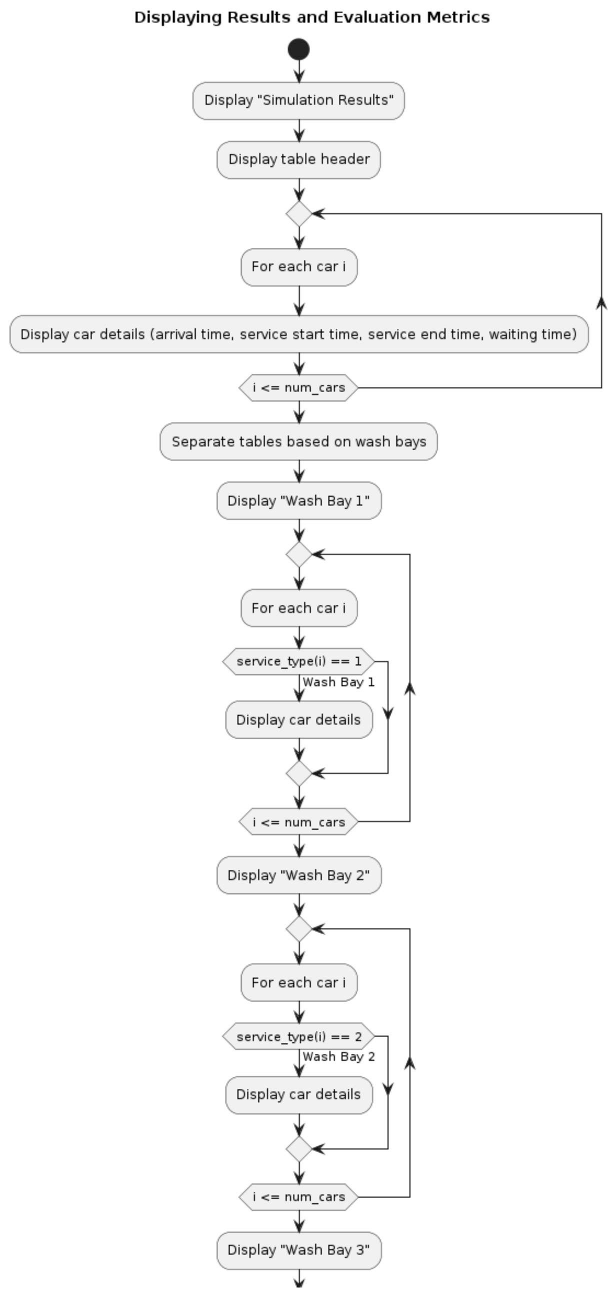
- Car Arrival and Service Simulation

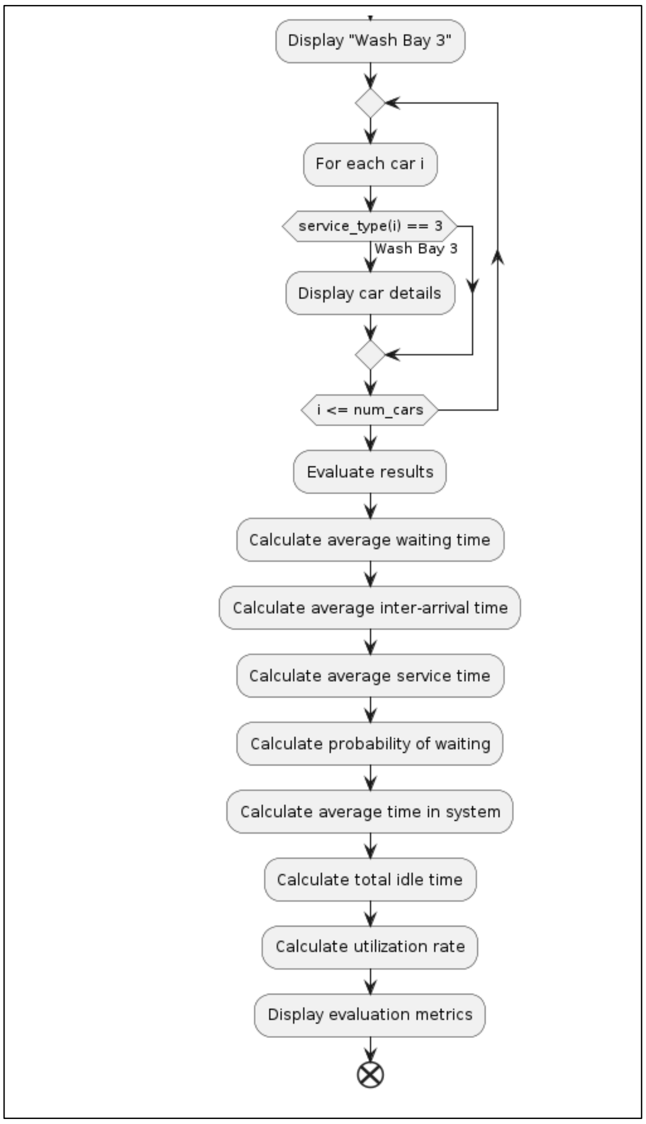

Car Arrival and Service Simulation
Initialize arrays for times and service types
For each car i
First Car
i == 1
Subsequent Cars
Set arrival time to 0
Determine inter-arrival time
Calculate arrival time
Determine service type and time
First Car
i == 1
Subsequent Cars
Set service start time to arrival time
Calculate service start time
Calculate service end time
Calculate waiting time
Display arrival, service start, and departure messages
i <= num_cars

- **Start**: The process begins by initializing arrays for storing times and service types.
- **Loop Through Each Car**: For each car in the simulation:
  - **First Car**:
    - If it is the first car, set its arrival time to 0.
  - **Subsequent Cars**:
    - For subsequent cars, determine the inter-arrival time using the generated random numbers.
    - Calculate the arrival time based on the previous car's arrival time and the inter-arrival time.
  - **Determine Service Type and Time**: Use the generated random numbers to determine the service type and the corresponding service time for the car.
  - **First Car Service Start**:
    - If it is the first car, set its service start time to its arrival time.
  - **Subsequent Cars Service Start**:
    - For subsequent cars, calculate the service start time as the maximum of the car's arrival time and the previous car's service end time.
  - **Calculate Service End Time**: Calculate the service end time based on the service start time and the service duration.
  - **Calculate Waiting Time**: Calculate the waiting time as the difference between the service start time and the arrival time.
  - **Display Messages**: Display messages indicating the car's arrival, service start, and departure times.

- **End**: The loop continues until all cars have been processed.

- Displaying Results and Evaluation Metrics




Display "Wash Bay 3"
For each car i
service_type(i) == 3
Wash Bay 3
Display car details
i <= num_cars
Evaluate results
Calculate average waiting time
Calculate average inter-arrival time
Calculate average service time
Calculate probability of waiting
Calculate average time in system
Calculate total idle time
Calculate utilization rate
Display evaluation metrics

- **Start**: The process begins by displaying "Simulation Results."
- **Display Table Header**: Display the header for the results table.
- **Loop Through Each Car**: For each car, display its arrival time, service start time, service end time, and waiting time.
- **Separate Tables by Wash Bays**:

  - **Display Wash Bay 1 Results**: Display the details for cars serviced at Wash Bay 1.
  - **Display Wash Bay 2 Results**: Display the details for cars serviced at Wash Bay 2.
  - **Display Wash Bay 3 Results**: Display the details for cars serviced at Wash Bay 3.

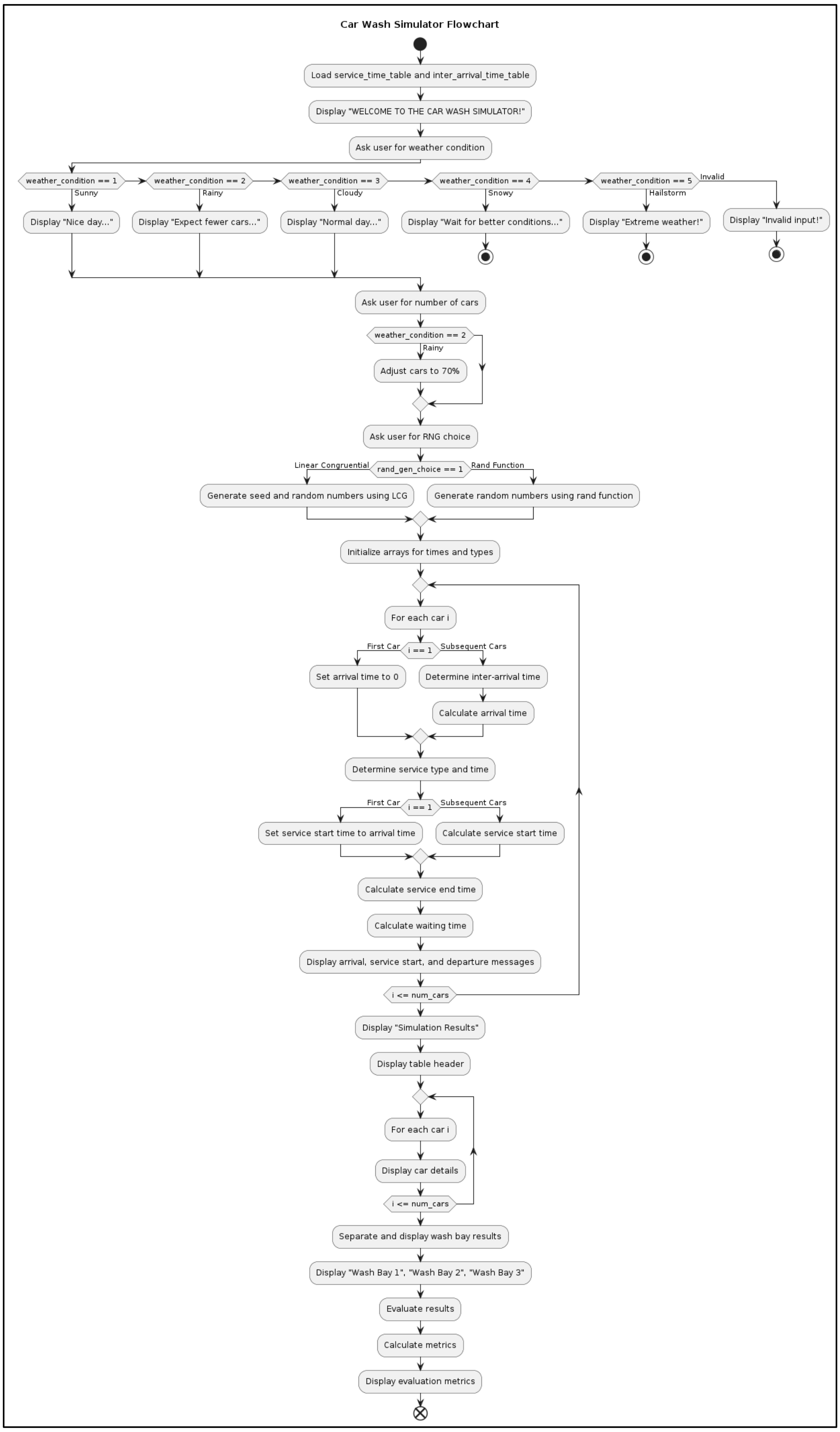
- **Evaluate Results**: Calculate various metrics:

  - **Average Waiting Time**: Calculate the average waiting time for all cars.

    - **Average Inter-Arrival Time**: Calculate the average inter-arrival time.
    - **Average Service Time**: Calculate the average service time.
    - **Probability of Waiting**: Calculate the probability that a car has to wait.
    - **Average Time in System**: Calculate the average time each car spends in the system.
    - **Total Idle Time**: Calculate the total idle time of the system.
    - **Utilization Rate**: Calculate the utilization rate of the wash bays.

- **Display Evaluation Metrics**: Display the calculated evaluation metrics.
- **End**: The process concludes after displaying all results and evaluation metrics.

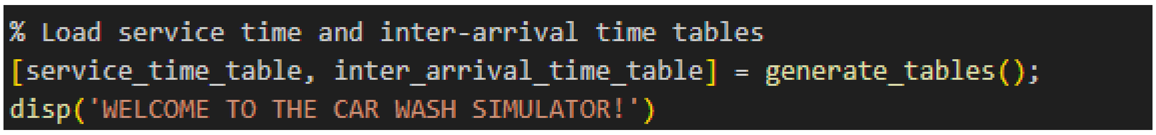
GENERAL FLOWCHART


Car Wash Simulator Flowchart
Load service_time_table and inter_arrival_time_table
Display "WELCOME TO THE CAR WASH SIMULATOR!"
Ask user for weather condition
weather_condition == 1
Sunny
Display "Nice day..."
weather_condition == 2
Rainy
Display "Expect fewer cars..."
weather_condition == 3
Cloudy
Display "Normal day..."
weather_condition == 4
Snowy
Display "Wait for better conditions..."
weather_condition == 5
Hailstorm
Display "Extreme weather!"
Invalid
Display "Invalid input!"
Ask user for number of cars
weather_condition == 2
Rainy
Adjust cars to 70%
Ask user for RNG choice
Linear Congruential
rand_gen_choice == 1
Rand Function
Generate seed and random numbers using LCG
Generate random numbers using rand function
Initialize arrays for times and types
For each car i
First Car
i == 1
Subsequent Cars
Set arrival time to 0
Determine inter-arrival time
Calculate arrival time
Determine service type and time
First Car
i == 1
Subsequent Cars
Set service start time to arrival time
Calculate service start time
Calculate service end time
Calculate waiting time
Display arrival, service start, and departure messages
i <= num_cars
Display "Simulation Results"
Display table header
For each car i
Display car details
i <= num_cars
Separate and display wash bay results
Display "Wash Bay 1", "Wash Bay 2", "Wash Bay 3"
Evaluate results
Calculate metrics
Display evaluation metrics

<u>EXPLANATION OF EACH LINES IN CODES</u>

car_wash_simulator.m

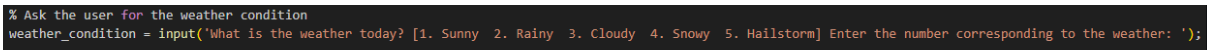
Initialization and Greeting



```
% Load service time and inter-arrival time tables
[service_time_table, inter_arrival_time_table] = generate_tables();
disp('WELCOME TO THE CAR WASH SIMULATOR!')
```

The script starts by calling the generate_tables() function, which loads or creates tables containing service times and inter-arrival times. Then, a welcome message is displayed to introduce the user to the car wash simulator.

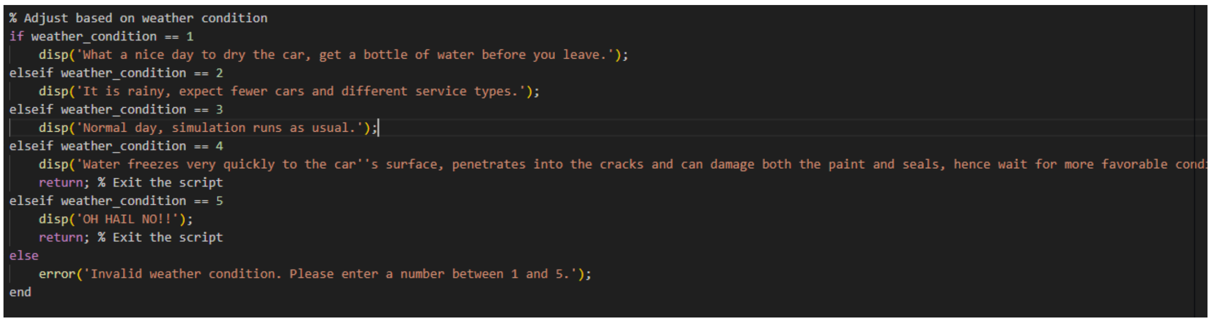
Input: Weather Condition



```
% Ask the user for the weather condition
weather_condition = input('What is the weather today? [1. Sunny  2. Rainy  3. Cloudy  4. Snowy  5. Hailstorm] Enter the number corresponding to the weather: ');
```

The script prompts the user to input the current weather condition by entering a number from 1 to 5, where each number corresponds to a specific weather type (Sunny, Rainy, Cloudy, Snowy, Hailstorm).

Weather-Based Adjustments

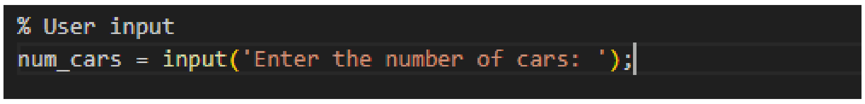


```
% Adjust based on weather condition
if weather_condition == 1
    disp('What a nice day to dry the car, get a bottle of water before you leave.');
elseif weather_condition == 2
    disp('It is rainy, expect fewer cars and different service types.');
elseif weather_condition == 3
    disp('Normal day, simulation runs as usual.');
elseif weather_condition == 4
    disp('Water freezes very quickly to the car''s surface, penetrates into the cracks and can damage both the paint and seals, hence wait for more favorable cond
    return; % Exit the script
elseif weather_condition == 5
    disp('OH HAIL NO!!');
    return; % Exit the script
else
    error('Invalid weather condition. Please enter a number between 1 and 5.');
end
```

The script checks the weather condition and displays an appropriate message for each type:

- **Sunny**: Displays a positive message about drying the car.
- **Rainy**: Warns about expecting fewer cars and different service types.
- **Cloudy**: States that the simulation runs as usual.
- **Snowy**: Warns about potential damage due to freezing and exits the script.
- **Hailstorm**: Displays a humorous message and exits the script.
- **Invalid Input**: Throws an error if the input is not between 1 and 5.

Input: Number of Cars



```
% User input
num_cars = input('Enter the number of cars: ');
```

The script asks the user to input the number of cars to be simulated in the car wash.

Adjust Number of Cars for Rainy Weather

```matlab
% Adjust the number of cars based on the weather condition
if weather_condition == 2
    adjusted_num_cars = max(1, floor(num_cars * 0.7)); % Reduce the number of cars to 70% if it is rainy
    fprintf('Due to the rainy season, only %d cars are allowed in.\n', adjusted_num_cars);
    num_cars = adjusted_num_cars;
end
```

If the weather is rainy, the script reduces the number of cars by 30% (to 70% of the original number) to reflect the reduced traffic in rainy conditions. It ensures that at least one car is present and updates the number of cars. EX: if user inputs 20 cars only 14 cars are allowed in.

Input: Random Number Generator Choice

```matlab
rand_gen_choice = input('Choose random number generator (1 for Linear Congruential, 2 for Rand): ');
```

The script prompts the user to choose a random number generator method: either Linear Congruential (1) or MATLAB's built-in Rand function (2).

Validate and Generate Random Numbers

```matlab
% Generate random numbers
if rand_gen_choice == 1
    seed = floor(rand() * 100); % Corrected syntax for generating a seed
    a = 1664525;
    c = 1013904223;
    m = 2^32;
    rand_numbers = rand_generator(seed, a, c, m, num_cars * 2); % Generate more random numbers for inter-arrival and service times
else
    rand_numbers = rand(1, num_cars * 2); % Generate more random numbers for inter-arrival and service times
end
```

The script validates the random number generator choice. If the choice is invalid, it throws an error. If the choice is valid:

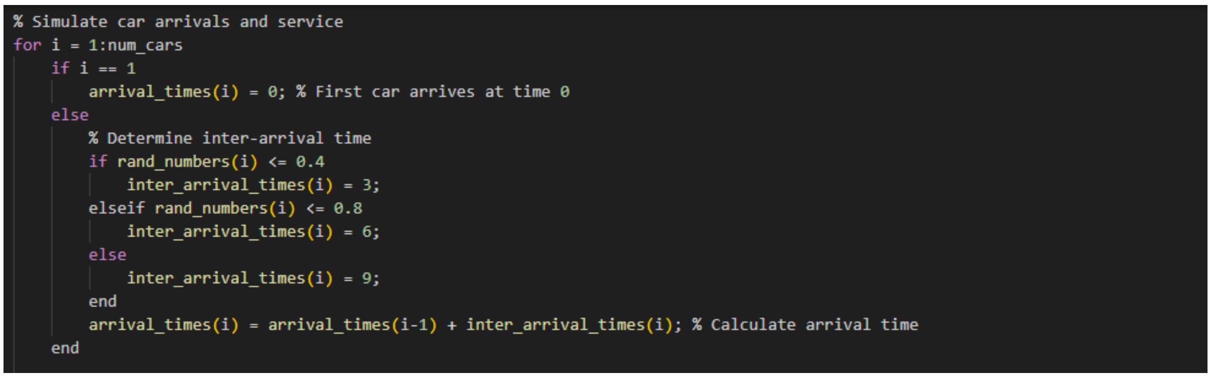
- For Linear Congruential: It initializes parameters for the generator and generates random numbers.
- For MATLAB's Rand: It generates random numbers using the built-in function.
- The random numbers are generated for both inter-arrival and service times.

Initialize Arrays

```matlab
% Initialize arrays
inter_arrival_times = zeros(1, num_cars);
arrival_times = zeros(1, num_cars);
service_times = zeros(1, num_cars);
waiting_times = zeros(1, num_cars);
service_start = zeros(1, num_cars);
service_end = zeros(1, num_cars);
service_type = zeros(1, num_cars);
```

The script initializes arrays to store the inter-arrival times, arrival times, service times, waiting times, service start times, service end times, and service types for each car.

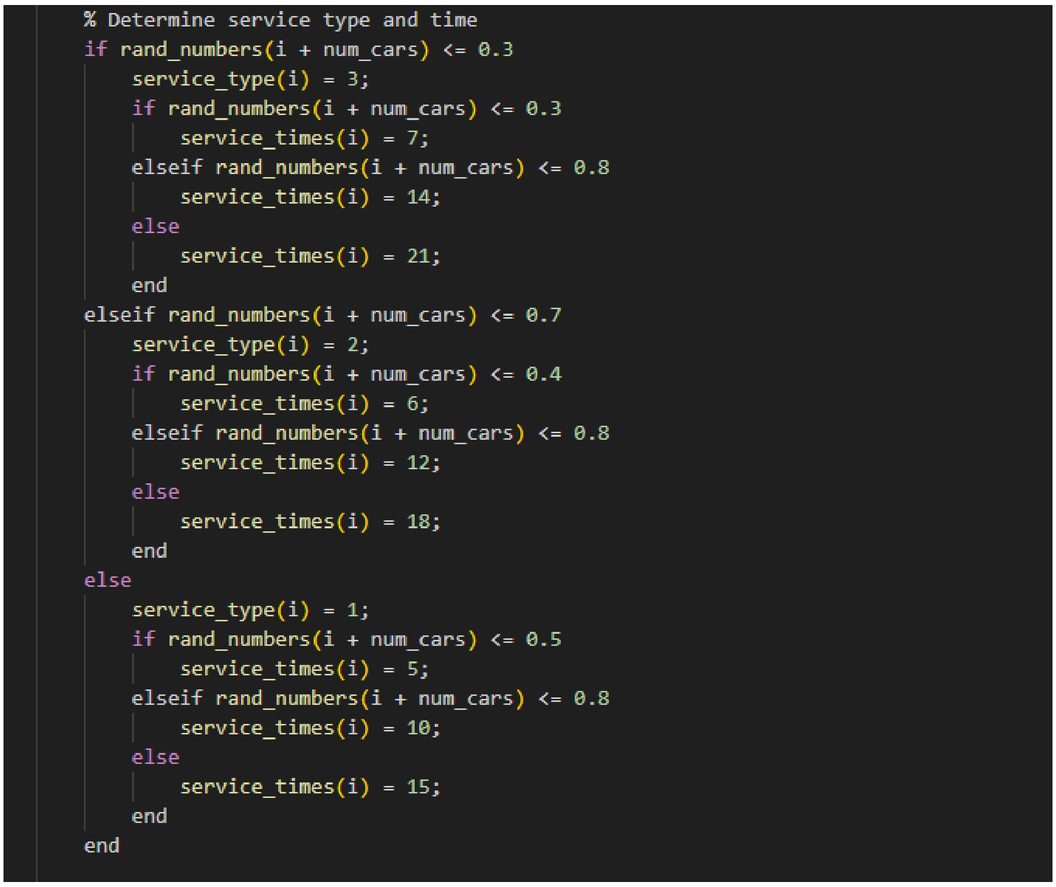
Simulate Car Arrivals



```matlab
% Simulate car arrivals and service
for i = 1:num_cars
    if i == 1
        arrival_times(i) = 0; % First car arrives at time 0
    else
        % Determine inter-arrival time
        if rand_numbers(i) <= 0.4
            inter_arrival_times(i) = 3;
        elseif rand_numbers(i) <= 0.8
            inter_arrival_times(i) = 6;
        else
            inter_arrival_times(i) = 9;
        end
        arrival_times(i) = arrival_times(i-1) + inter_arrival_times(i); % Calculate arrival time
    end
```

The script starts a loop that processes each car from 1 to the total number of cars. For the first car (i == 1), it sets the arrival time to 0, meaning it arrives at the start of the simulation. For every other car, the script determines the inter-arrival time based on the generated random numbers: if the random number is less than or equal to 0.4, the inter-arrival time is set to 3 minutes; if the random number is between 0.4 and 0.8, the inter-arrival time is set to 6 minutes; if the random number is greater than 0.8, the inter-arrival time is set to 9 minutes. The arrival time for each subsequent car is then calculated by adding the inter-arrival time to the arrival time of the previous car.

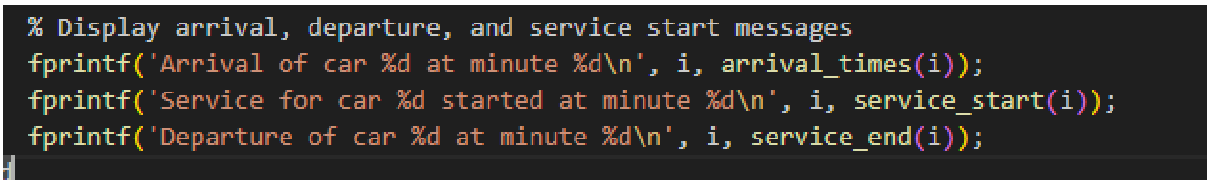
Service Times and Types



```matlab
    % Determine service type and time
    if rand_numbers(i + num_cars) <= 0.3
        service_type(i) = 3;
        if rand_numbers(i + num_cars) <= 0.3
            service_times(i) = 7;
        elseif rand_numbers(i + num_cars) <= 0.8
            service_times(i) = 14;
        else
            service_times(i) = 21;
        end
    elseif rand_numbers(i + num_cars) <= 0.7
        service_type(i) = 2;
        if rand_numbers(i + num_cars) <= 0.4
            service_times(i) = 6;
        elseif rand_numbers(i + num_cars) <= 0.8
            service_times(i) = 12;
        else
            service_times(i) = 18;
        end
    else
        service_type(i) = 1;
        if rand_numbers(i + num_cars) <= 0.5
            service_times(i) = 5;
        elseif rand_numbers(i + num_cars) <= 0.8
            service_times(i) = 10;
        else
            service_times(i) = 15;
        end
    end
```

For each car, the script determines the service type and corresponding service time based on random numbers: if the random number is less than or equal to 0.3, the service type is set to 3, with a service time of 7, 14, or 21 minutes; if the random number is between 0.3 and 0.7, the service type is set to 2, with a service time of 6, 12, or 18 minutes; and if the random number is greater than 0.7, the service type is set to 1, with a service time of 5, 10, or 15 minutes. For the first car, the service start time is the same as its arrival time. For subsequent cars, the service start time is the maximum of the car's arrival time and the service end time of the previous car. The service end time is then calculated by adding the service time to the service start time. The waiting time for each car is calculated as the difference between its service start time and arrival time.

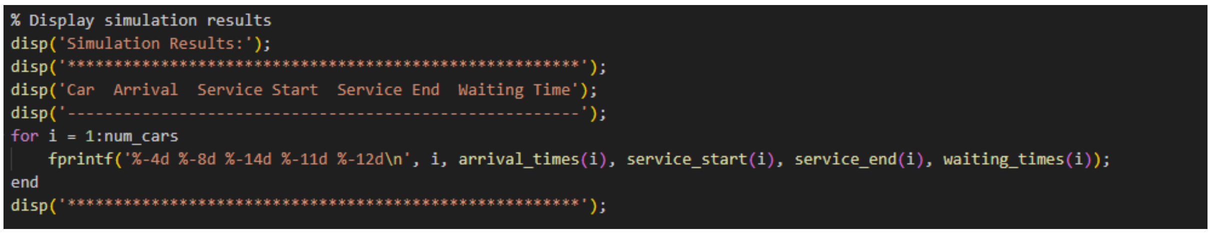
Display arrival, departure and service start messages



```
% Display arrival, departure, and service start messages
fprintf('Arrival of car %d at minute %d\n', i, arrival_times(i));
fprintf('Service for car %d started at minute %d\n', i, service_start(i));
fprintf('Departure of car %d at minute %d\n', i, service_end(i));
```

This segment of the code prints messages about each car's key events during the simulation. It uses fprintf to display the arrival time, service start time, and departure time for each car (i). The arrival time is taken from arrival_times(i), the service start time from service_start(i), and the departure time from service_end(i). These messages help track each car's progress through the simulation, providing a clear log of when each car arrives, starts service, and departs.

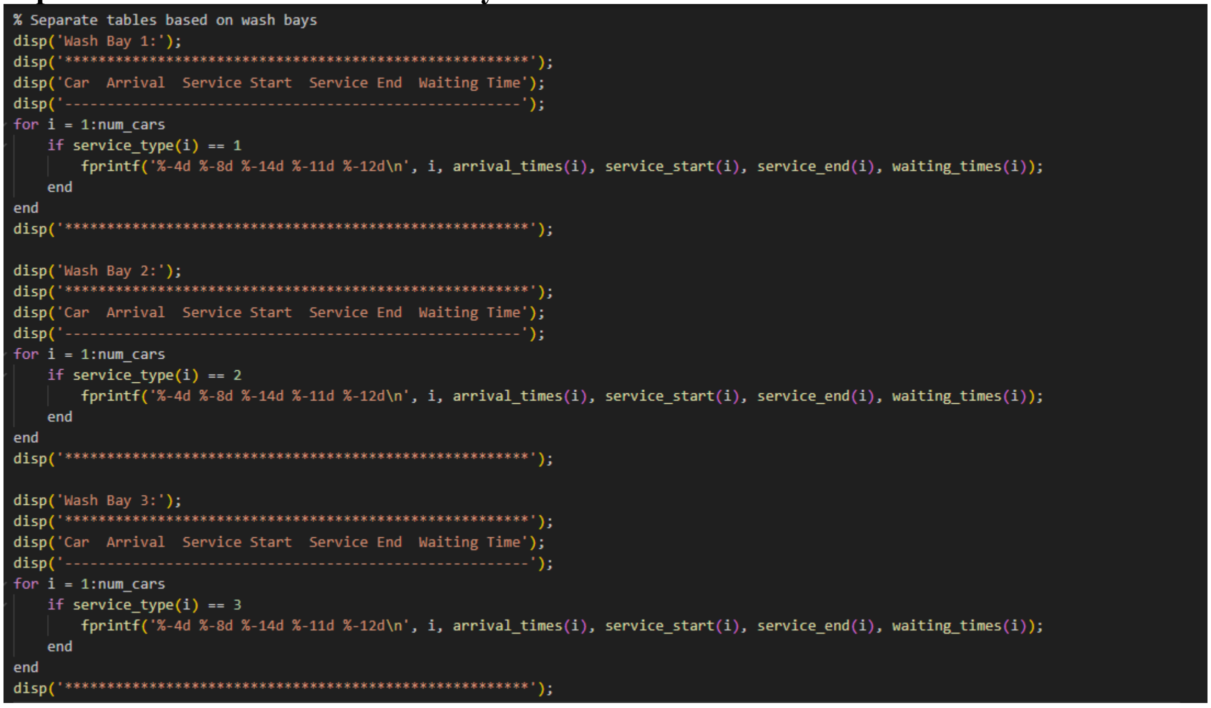
Display Simulation Results



```
% Display simulation results
disp('Simulation Results:');
disp('*******************************************************');
disp('Car  Arrival  Service Start  Service End  Waiting Time');
disp('-------------------------------------------------------');
for i = 1:num_cars
    fprintf('%-4d %-8d %-14d %-11d %-12d\n', i, arrival_times(i), service_start(i), service_end(i), waiting_times(i));
end
disp('*******************************************************');
```

The script displays the simulation results in a formatted table showing each car's arrival time, service start time, service end time, and waiting time.

Separate Tables Based on Wash Bays



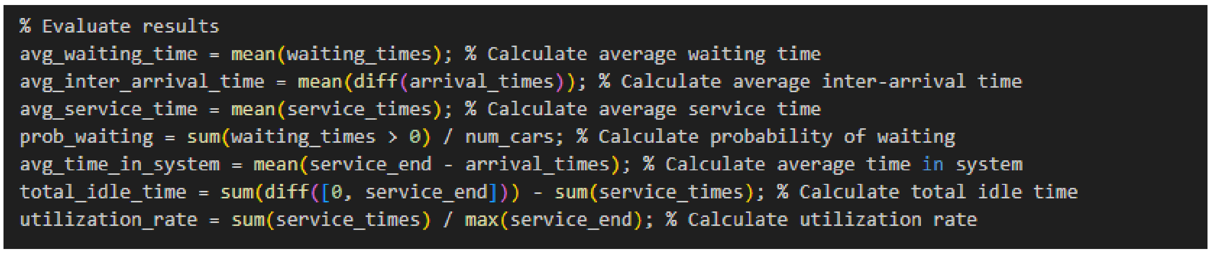
```
% Separate tables based on wash bays
disp('Wash Bay 1:');
disp('******************************************************');
disp('Car  Arrival  Service Start  Service End  Waiting Time');
disp('-----------------------------------------------------');
for i = 1:num_cars
    if service_type(i) == 1
        fprintf('%-4d %-8d %-14d %-11d %-12d\n', i, arrival_times(i), service_start(i), service_end(i), waiting_times(i));
    end
end
disp('******************************************************');

disp('Wash Bay 2:');
disp('******************************************************');
disp('Car  Arrival  Service Start  Service End  Waiting Time');
disp('-----------------------------------------------------');
for i = 1:num_cars
    if service_type(i) == 2
        fprintf('%-4d %-8d %-14d %-11d %-12d\n', i, arrival_times(i), service_start(i), service_end(i), waiting_times(i));
    end
end
disp('******************************************************');

disp('Wash Bay 3:');
disp('******************************************************');
disp('Car  Arrival  Service Start  Service End  Waiting Time');
disp('------------------------------------------------------');
for i = 1:num_cars
    if service_type(i) == 3
        fprintf('%-4d %-8d %-14d %-11d %-12d\n', i, arrival_times(i), service_start(i), service_end(i), waiting_times(i));
    end
end
disp('******************************************************');
```

The script separates and displays the results for each wash bay based on the service types (1, 2, or 3).

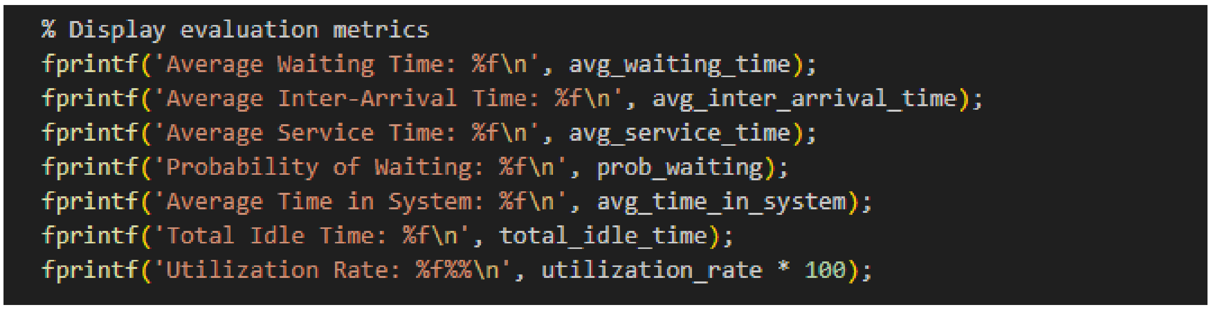
Evaluate Results



```
% Evaluate results
avg_waiting_time = mean(waiting_times); % Calculate average waiting time
avg_inter_arrival_time = mean(diff(arrival_times)); % Calculate average inter-arrival time
avg_service_time = mean(service_times); % Calculate average service time
prob_waiting = sum(waiting_times > 0) / num_cars; % Calculate probability of waiting
avg_time_in_system = mean(service_end - arrival_times); % Calculate average time in system
total_idle_time = sum(diff([0, service_end])) - sum(service_times); % Calculate total idle time
utilization_rate = sum(service_times) / max(service_end); % Calculate utilization rate
```

The script calculates various performance metrics, including the average waiting time, which is the mean waiting time for all cars; the average inter-arrival time, which is the mean time between car arrivals; and the average service time, which is the mean service time for all cars. It also calculates the probability of waiting, which is the proportion of cars that had to wait; the average time in the system, which is the mean time a car spends from arrival to departure; the total idle time, which is the total time when no car is being serviced; and the utilization rate, which is the proportion of time the system is being utilized.

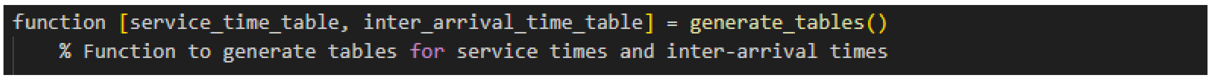
Display Evaluation Metrics



```
% Display evaluation metrics
fprintf('Average Waiting Time: %f\n', avg_waiting_time);
fprintf('Average Inter-Arrival Time: %f\n', avg_inter_arrival_time);
fprintf('Average Service Time: %f\n', avg_service_time);
fprintf('Probability of Waiting: %f\n', prob_waiting);
fprintf('Average Time in System: %f\n', avg_time_in_system);
fprintf('Total Idle Time: %f\n', total_idle_time);
fprintf('Utilization Rate: %f%%\n', utilization_rate * 100);
```

Finally, the script displays the calculated performance metrics in a readable format.

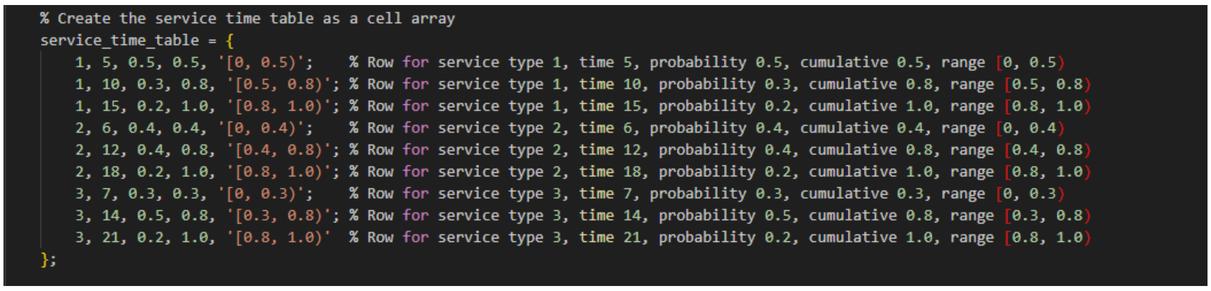
generate_tables.m



```
function [service_time_table, inter_arrival_time_table] = generate_tables()
    % Function to generate tables for service times and inter-arrival times
```

This defines a function named generate_tables that returns two outputs: service_time_table and inter_arrival_time_table.

Service Time Table

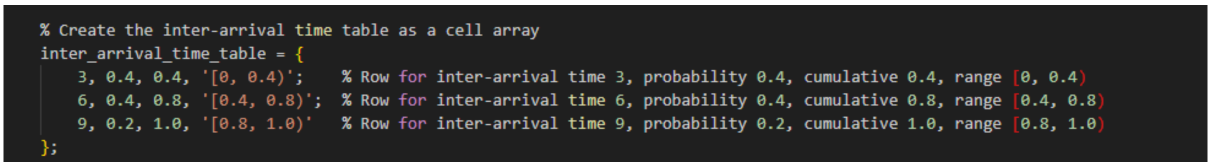


```
% Create the service time table as a cell array
service_time_table = {
    1, 5, 0.5, 0.5, '[0, 0.5)';     % Row for service type 1, time 5, probability 0.5, cumulative 0.5, range [0, 0.5)
    1, 10, 0.3, 0.8, '[0.5, 0.8)'; % Row for service type 1, time 10, probability 0.3, cumulative 0.8, range [0.5, 0.8)
    1, 15, 0.2, 1.0, '[0.8, 1.0)'; % Row for service type 1, time 15, probability 0.2, cumulative 1.0, range [0.8, 1.0)
    2, 6, 0.4, 0.4, '[0, 0.4)';     % Row for service type 2, time 6, probability 0.4, cumulative 0.4, range [0, 0.4)
    2, 12, 0.4, 0.8, '[0.4, 0.8)'; % Row for service type 2, time 12, probability 0.4, cumulative 0.8, range [0.4, 0.8)
    2, 18, 0.2, 1.0, '[0.8, 1.0)'; % Row for service type 2, time 18, probability 0.2, cumulative 1.0, range [0.8, 1.0)
    3, 7, 0.3, 0.3, '[0, 0.3)';     % Row for service type 3, time 7, probability 0.3, cumulative 0.3, range [0, 0.3)
    3, 14, 0.5, 0.8, '[0.3, 0.8)'; % Row for service type 3, time 14, probability 0.5, cumulative 0.8, range [0.3, 0.8)
    3, 21, 0.2, 1.0, '[0.8, 1.0)'  % Row for service type 3, time 21, probability 0.2, cumulative 1.0, range [0.8, 1.0)
};
```

This section creates a cell array named service_time_table which contains information about different service types and their associated times, probabilities, cumulative probabilities, and ranges.

- **First Column**: Service type (1, 2, or 3).
- **Second Column**: Service time (e.g., 5, 10, 15 minutes).
- **Third Column**: Probability of that service time.
- **Fourth Column**: Cumulative probability up to that service time.
- **Fifth Column**: Range of random numbers that correspond to that service time.

Inter-Arrival Time Table



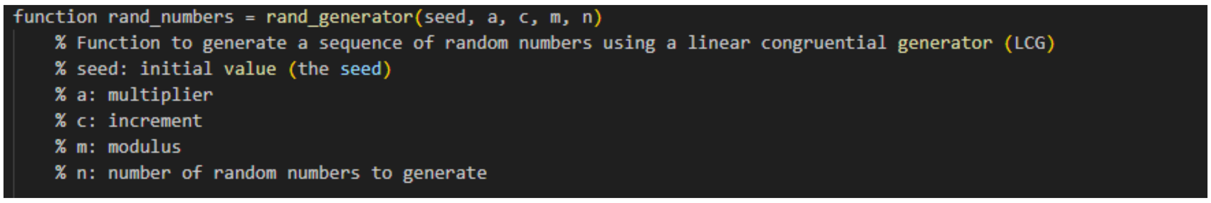
```
% Create the inter-arrival time table as a cell array
inter_arrival_time_table = {
    3, 0.4, 0.4, '[0, 0.4)';    % Row for inter-arrival time 3, probability 0.4, cumulative 0.4, range [0, 0.4)
    6, 0.4, 0.8, '[0.4, 0.8)';  % Row for inter-arrival time 6, probability 0.4, cumulative 0.8, range [0.4, 0.8)
    9, 0.2, 1.0, '[0.8, 1.0)'   % Row for inter-arrival time 9, probability 0.2, cumulative 1.0, range [0.8, 1.0)
};
```

This section creates a cell array named inter_arrival_time_table which contains information about inter-arrival times, their associated probabilities, cumulative probabilities, and ranges.

- **First Column**: Inter-arrival time (e.g., 3, 6, 9 minutes).
- **Second Column**: Probability of that inter-arrival time.
- **Third Column**: Cumulative probability up to that inter-arrival time.
- **Fourth Column**: Range of random numbers that correspond to that inter-arrival time.

RAND_GENERATOR

Function Definition

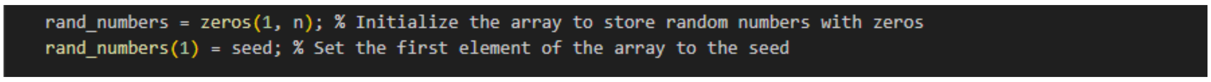


```
function rand_numbers = rand_generator(seed, a, c, m, n)
    % Function to generate a sequence of random numbers using a linear congruential generator (LCG)
    % seed: initial value (the seed)
    % a: multiplier
    % c: increment
    % m: modulus
    % n: number of random numbers to generate
```

This defines a function named rand_generator that generates a sequence of random numbers using the Linear Congruential Generator (LCG) method. The function takes five inputs:

- ‘seed’: The initial value or starting point for the random number generation.
- a: The multiplier used in the LCG formula.
- c: The increment added in the LCG formula.
- m: The modulus that limits the range of generated numbers.
- n: The number of random numbers to generate.

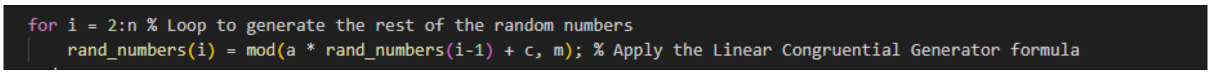
Initialize Array



```
    rand_numbers = zeros(1, n); % Initialize the array to store random numbers with zeros
    rand_numbers(1) = seed; % Set the first element of the array to the seed
```

The script initializes an array named rand_numbers with n elements, all set to zero. The first element of this array is then set to the seed value provided as input.

Loop to Generate Random Numbers



```
  for i = 2:n % Loop to generate the rest of the random numbers
      rand_numbers(i) = mod(a * rand_numbers(i-1) + c, m); % Apply the Linear Congruential Generator formula
```

This loop runs from 2 to n, generating the random numbers using the LCG formula:

 LCG Formula

$$X_{n+1} = (a \cdot X_n + c) \mod m$$

Where:

- X is the sequence of pseudo-random values,
- a is the multiplier,
- c is the increment,
- m is the modulus,
- X_0 is the initial value (the seed).

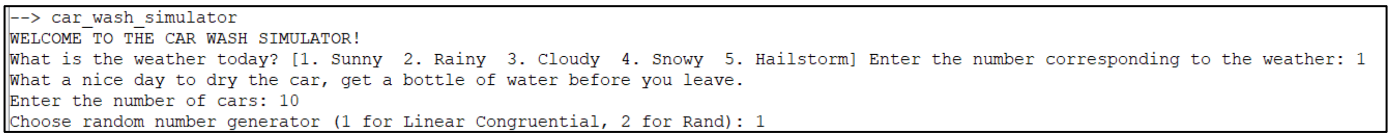
 This formula is applied iteratively to generate the sequence of random numbers.

Normalize Random Numbers

```
rand_numbers = rand_numbers / m; % Normalize the random numbers to be between 0 and 1
```

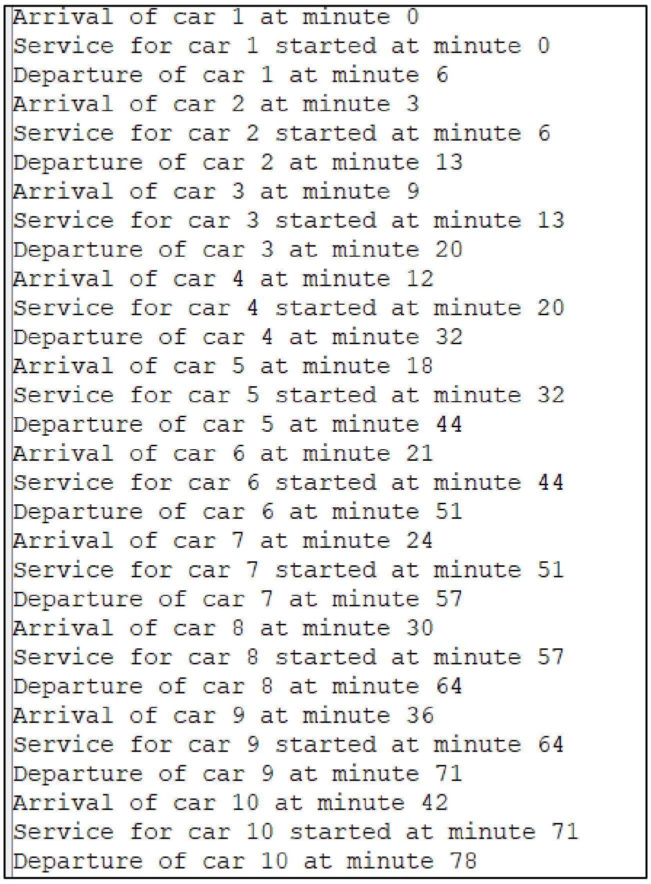
After generating the sequence of random numbers, the script normalizes them by dividing each element in the rand_numbers array by the modulus (m). This ensures that the random numbers are scaled to be within the range of 0 to 1.

<u>EXAMPLE OUTPUT</u>



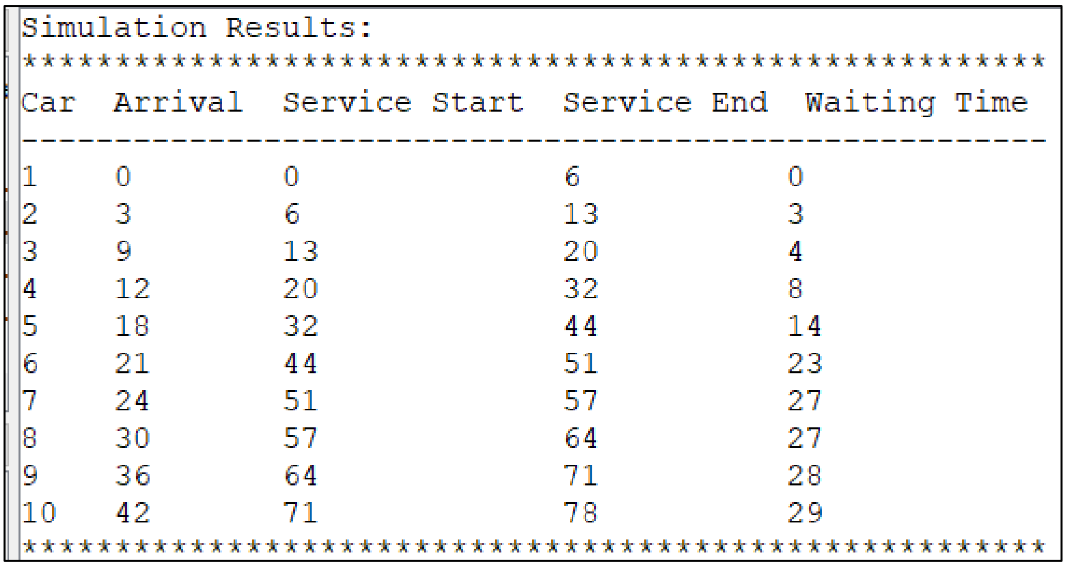
```
--> car_wash_simulator
WELCOME TO THE CAR WASH SIMULATOR!
What is the weather today? [1. Sunny  2. Rainy  3. Cloudy  4. Snowy  5. Hailstorm] Enter the number corresponding to the weather: 1
What a nice day to dry the car, get a bottle of water before you leave.
Enter the number of cars: 10
Choose random number generator (1 for Linear Congruential, 2 for Rand): 1
```

- Entered car_wash_simulator to initiate the simulator. (Make sure all the .m files are in the same directory).
- Entered 1 for the weather choice because to show normality.
- 10 cars entered.
- Random Number generator selection (1 selected for LCG uses the formula).



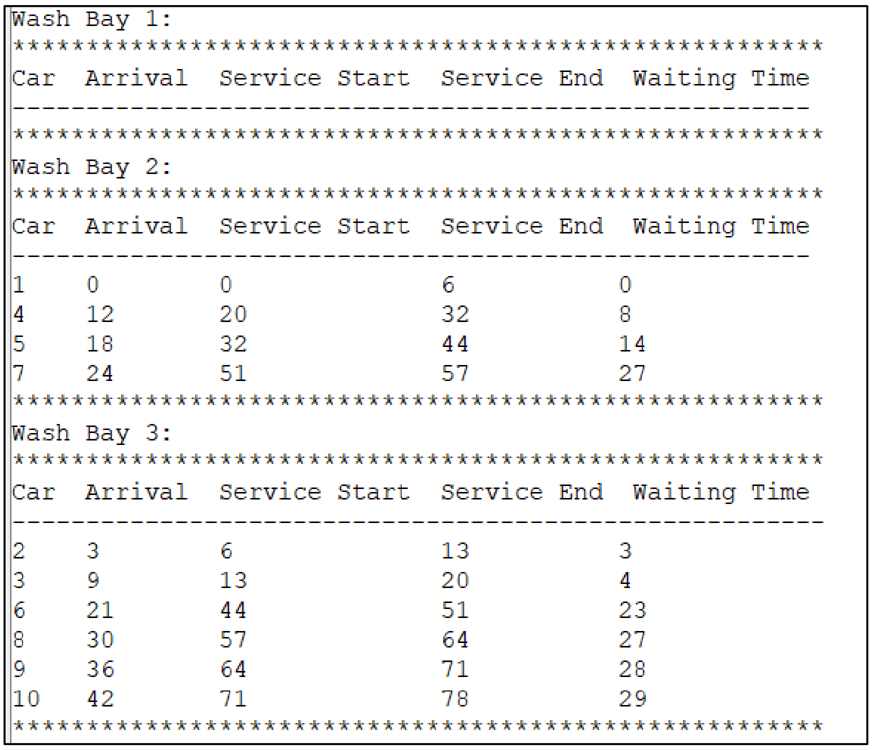
```
Arrival of car 1 at minute 0
Service for car 1 started at minute 0
Departure of car 1 at minute 6
Arrival of car 2 at minute 3
Service for car 2 started at minute 6
Departure of car 2 at minute 13
Arrival of car 3 at minute 9
Service for car 3 started at minute 13
Departure of car 3 at minute 20
Arrival of car 4 at minute 12
Service for car 4 started at minute 20
Departure of car 4 at minute 32
Arrival of car 5 at minute 18
Service for car 5 started at minute 32
Departure of car 5 at minute 44
Arrival of car 6 at minute 21
Service for car 6 started at minute 44
Departure of car 6 at minute 51
Arrival of car 7 at minute 24
Service for car 7 started at minute 51
Departure of car 7 at minute 57
Arrival of car 8 at minute 30
Service for car 8 started at minute 57
Departure of car 8 at minute 64
Arrival of car 9 at minute 36
Service for car 9 started at minute 64
Departure of car 9 at minute 71
Arrival of car 10 at minute 42
Service for car 10 started at minute 71
Departure of car 10 at minute 78
```

- Arrival, service time and leaving time of each car are generated randomly.



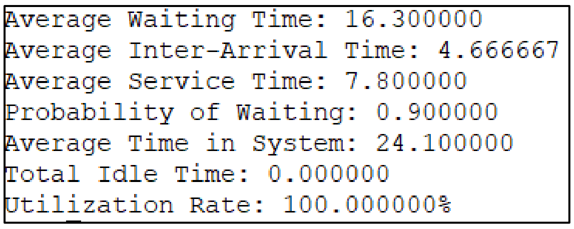
```
Simulation Results:
*********************************************************
Car  Arrival  Service Start  Service End  Waiting Time
---------------------------------------------------------
1    0        0              6            0
2    3        6              13           3
3    9        13             20           4
4    12       20             32           8
5    18       32             44           14
6    21       44             51           23
7    24       51             57           27
8    30       57             64           27
9    36       64             71           28
10   42       71             78           29
*********************************************************
```

- Shows entire simulation result table for all the cars.



```
Wash Bay 1:
*******************************************************
Car  Arrival  Service Start  Service End  Waiting Time
------------------------------------------------------
*******************************************************
Wash Bay 2:
*******************************************************
Car  Arrival  Service Start  Service End  Waiting Time
------------------------------------------------------
1    0        0              6            0
4    12       20             32           8
5    18       32             44           14
7    24       51             57           27
*******************************************************
Wash Bay 3:
*******************************************************
Car  Arrival  Service Start  Service End  Waiting Time
-------------------------------------------------------
2    3        6              13           3
3    9        13             20           4
6    21       44             51           23
8    30       57             64           27
9    36       64             71           28
10   42       71             78           29
*******************************************************
```

- Cars are separated to random wash bays from three (in this case, 4 cars went to wash bay 2, 6 went to the third wash bay and no cars went to the first).



```
Average Waiting Time: 16.300000
Average Inter-Arrival Time: 4.666667
Average Service Time: 7.800000
Probability of Waiting: 0.900000
Average Time in System: 24.100000
Total Idle Time: 0.000000
Utilization Rate: 100.000000%
```

-Evaluation results are shown.

Average Waiting Time

$$\frac{0+3+4+8+14+23+27+27+28+29}{10}=\frac{163}{10}=16.3$$

Average Inter-Arrival Time

Inter-arrival times:

$$(3,6,3,6,3,3,6,6,6)$$

Average Inter-Arrival Time:

$$\frac{3+6+3+6+3+3+6+6+6}{9}=\frac{42}{9}=4.67$$

Average Service Time

Service times:

$$(6,7,7,12,12,7,6,7,7,7)$$

Average Service Time:

$$\frac{6+7+7+12+12+7+6+7+7+7}{10}=\frac{78}{10}=7.8$$

Probability of Waiting

Number of cars that waited:

9 out of 10 (all but the first car)

Probability of Waiting:

$$\frac{9}{10}=0.9$$

Average Time in System

Total time in the system for each car:

$$(6, 10, 11, 20, 26, 30, 33, 34, 35, 36)$$

Average Time in System:

$$\frac{6 + 10 + 11 + 20 + 26 + 30 + 33 + 34 + 35 + 36}{10} = \frac{241}{10} = 24.1$$

Total Idle Time

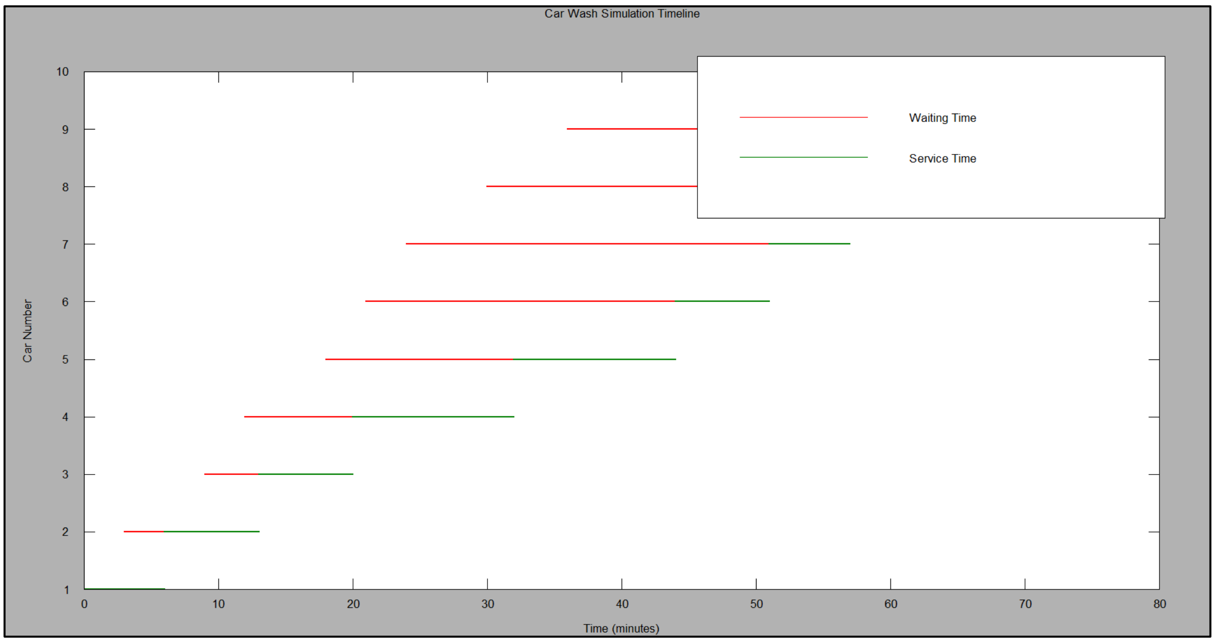
Given that the utilization rate is 100%, the idle time is zero.

Utilization Rate

Since the service bays are constantly in use:

$$\frac{TotalServiceTime}{TotalTime} = 100\%$$

- A sample graph is created .



The graph displayed is a **Gantt chart** style timeline that shows the waiting and service times for each car in the car wash simulation.

Explanation of the Graph

- **X-axis (Time in minutes)**: This axis represents the total time from the start of the simulation, marked in minutes.

- **Y-axis (Car Number)**: This axis represents the individual cars in the simulation, each identified by their car number (1 to 10 in this case).
- **Red Lines (Waiting Time)**: These lines represent the duration each car waited before the service started. The length of the red line indicates the amount of time the car waited.
- **Green Lines (Service Time)**: These lines represent the duration each car was being serviced. The length of the green line indicates the amount of time the car was under service.

Interpretation of the Graph

- **Car 1**: Arrived at time 0, started service immediately, and was serviced for 6 minutes (no waiting time).
- **Car 2**: Arrived at time 3, waited for 3 minutes (red line from 3 to 6), and was serviced for 7 minutes (green line from 6 to 13).
- **Car 3**: Arrived at time 9, waited for 4 minutes (red line from 9 to 13), and was serviced for 7 minutes (green line from 13 to 20).
- **Car 4**: Arrived at time 12, waited for 8 minutes (red line from 12 to 20), and was serviced for 12 minutes (green line from 20 to 32).
- **Car 5**: Arrived at time 18, waited for 14 minutes (red line from 18 to 32), and was serviced for 12 minutes (green line from 32 to 44).
- **Car 6**: Arrived at time 21, waited for 23 minutes (red line from 21 to 44), and was serviced for 7 minutes (green line from 44 to 51).
- **Car 7**: Arrived at time 24, waited for 27 minutes (red line from 24 to 51), and was serviced for 6 minutes (green line from 51 to 57).
- **Car 8**: Arrived at time 30, waited for 27 minutes (red line from 30 to 57), and was serviced for 7 minutes (green line from 57 to 64).
- **Car 9**: Arrived at time 36, waited for 28 minutes (red line from 36 to 64), and was serviced for 7 minutes (green line from 64 to 71).
- **Car 10**: Arrived at time 42, waited for 29 minutes (red line from 42 to 71), and was serviced for 7 minutes (green line from 71 to 78).

Purpose of the Graph

The graph visually represents how each car progresses through the car wash process, showing both the waiting time and the service time. This allows for a clear and immediate understanding of the efficiency of the service process, highlighting any potential bottlenecks or delays in the system.

SUMMARY

In this assignment, we successfully developed a comprehensive car wash simulator that models the queuing system for a car wash center with three wash bays. The simulator accounts for varying service types, inter-arrival times, and external factors such as weather conditions. Through the integration of a simple user interface for calibration, we provided users the ability to adjust probabilities and service times, ensuring the simulator's flexibility and accuracy.

We generated and analyzed the simulation data, which was then presented through detailed tables and graphical representations. The Gantt chart style timeline provided a clear visualization of the waiting and service times for each car, offering insights into the efficiency and performance of the car wash process.

The car wash simulator effectively models the dynamics of a car wash center, handling multiple variables and conditions to deliver accurate and insightful data. Key components and achievements of the assignment include:

1. **Data Generation**:
   - Developed custom tables for service times and inter-arrival times.
   - Incorporated a simple user interface for adjusting probabilities and ranges.
2. **Simulation System**:
   - Simulated car arrivals and services, accounting for different weather conditions.
   - Implemented efficient algorithms to manage the queuing and servicing processes.
3. **Report and Data Analysis**:
   - Provided detailed tables displaying arrival times, service start and end times, and waiting times for each car.
   - Evaluated key performance metrics such as average waiting time, average inter-arrival time, average service time, probability of waiting, and utilization rate.
4. **Graphical Representation**:
   - Created a Gantt chart style timeline to visually represent the waiting and service times, facilitating easy analysis and understanding of the simulation results.
5. **User Calibration**:
   - Enabled users to choose between different random number generators (Linear Congruential and Rand function).
   - Adjusted the number of cars based on weather conditions to reflect real-world scenarios.

Overall, the car wash simulator provides a robust and flexible tool for analyzing the performance of a car wash center. The detailed data and visual representations offer valuable insights into potential bottlenecks and areas for improvement, ensuring that the system can be optimized for better efficiency and customer satisfaction.

--THE END---