Faculty of Computing and Informatics (FCI)
Multimedia University
Cyberjaya

**CMA 6134 Computational Methods**

Trimester 2410

**Lecture Session: TC1L**
**Tutorial Session: TT3L**
**Submitted to: Tong Hau Lee**

**Group Member**


| ID | Name | Phone Number | Email |
|---|---|---|---|
| 1221102540 | CHIN ZHEN HO | 0182067278 | 1221102540@soffice.mmu.edu.my |
| 1221102007 | ERIC TEOH WEI XIANG | 0174063708 | 1221102007@soffice.mmu.edu.my |
| 1221101777 | BERNARD RYAN SIM KANG XUAN | 0167168548 | 1221101777@soffice.mmu.edu.my |
| 1221103201 | GAN SHAO YANG | 01154218068 | 1221103201@soffice.mmu.edu.my |

## User manual

1. Enter main to run our simulation.
2. Enter rand or lcg to choose the generator.
3. Enter Number of cars.

## Elaborate the details of simulation

The simulation aims to model a car wash system with three wash bays.
The following steps and functionalities are implemented:

**Random Number Generator Selection:**
The user is prompted to choose the type of random number generator (rand or lcg).

**Parameter Definition:**
Parameters for generating service times and inter-arrival times are defined, including their minimum and maximum values and the sequence length.

**Service Time Generation:**
Service times for the wash bays are generated along with their probabilities, cumulative distribution functions (CDFs), and random number ranges using the Random_Service_Time function.

**Inter-Arrival Time Generation:**
Inter-arrival times for the wash bays are generated along with their probabilities, cumulative distribution functions (CDFs), and random number ranges using the Inter_Arrival_Time function.

**Service Type Generation:**
Car wash service types (Basic Wash, Deluxe Wash, Premium Wash) are generated along with their probabilities, CDFs, and random number ranges using the Car_Wash_Service_Type function.

**Car Number and Attribute Generation:**
The sequence of car numbers and their associated random numbers is generated using the Generate_Car_Numbers function.

**Simulation Execution:**
The car wash process is simulated for the specified number of cars. The calculate_car_wash_table function processes each car, determining their service times, waiting times, and which wash bay they are assigned to.

**Result Calculation:**
The averages and probabilities of various metrics (waiting time, inter-arrival time, service time, etc.) are calculated and displayed using the calculate_averages function.

## Function Details:

**Random_Service_Time:**
Generates service times for three wash bays, their probabilities, CDFs, and random number ranges.
Uses the selected random number generator (rand or lcg).

**Inter_Arrival_Time:**
Generates inter-arrival times, their probabilities, CDFs, and random number ranges.
Ensures probabilities are within specified bounds.

**Car_Wash_Service_Type:**
Generates service types for the car wash, their probabilities, CDFs, and random number ranges.

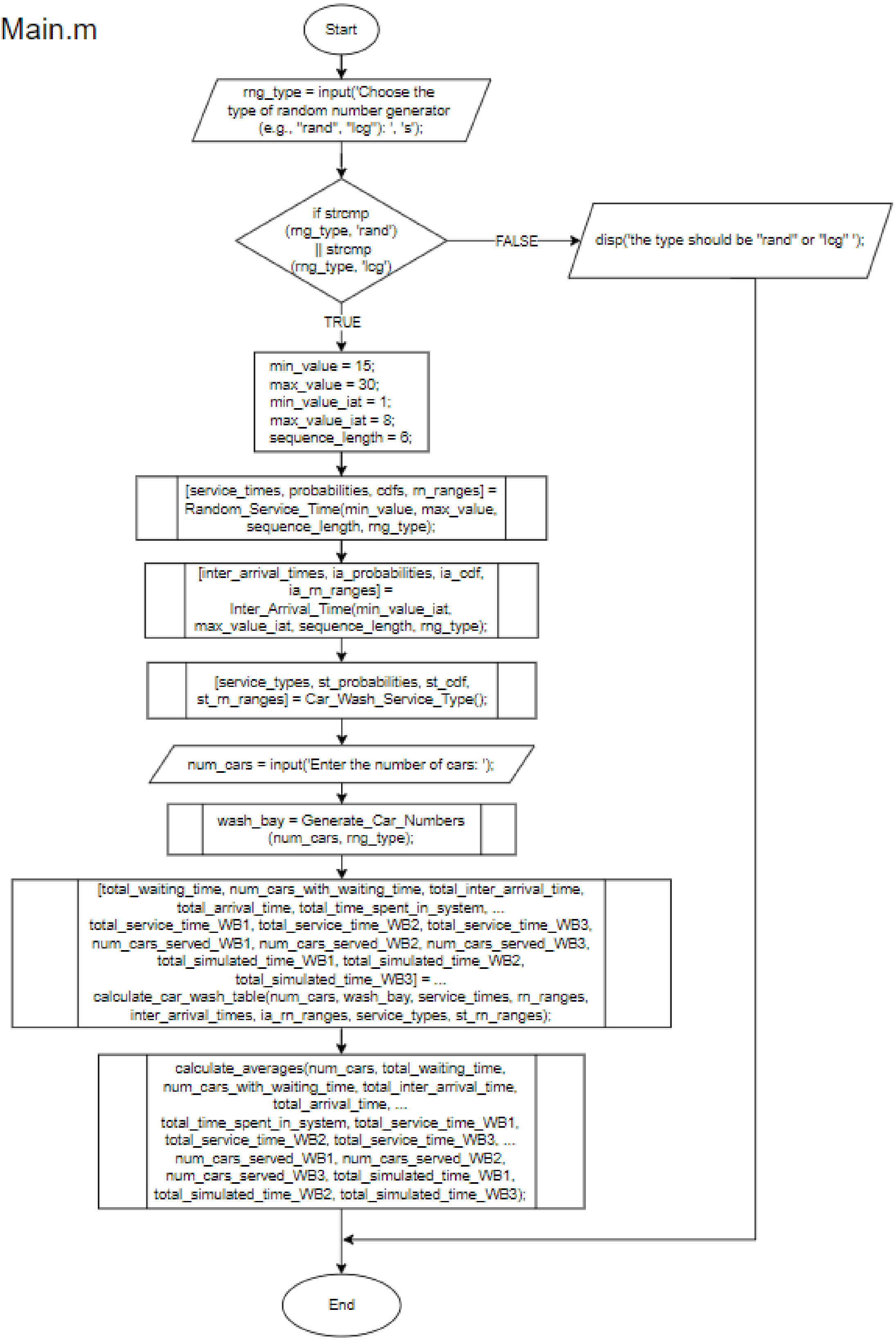
**Generate_Car_Numbers:**
Generates car numbers and their associated random numbers for inter-arrival time, service time, and service type.

**calculate_car_wash_table:**
Processes each car to determine their inter-arrival time, arrival time, service type, service time, waiting time, and the time spent in the system.
Assigns cars to wash bays based on their availability and calculates the end times for each wash bay.

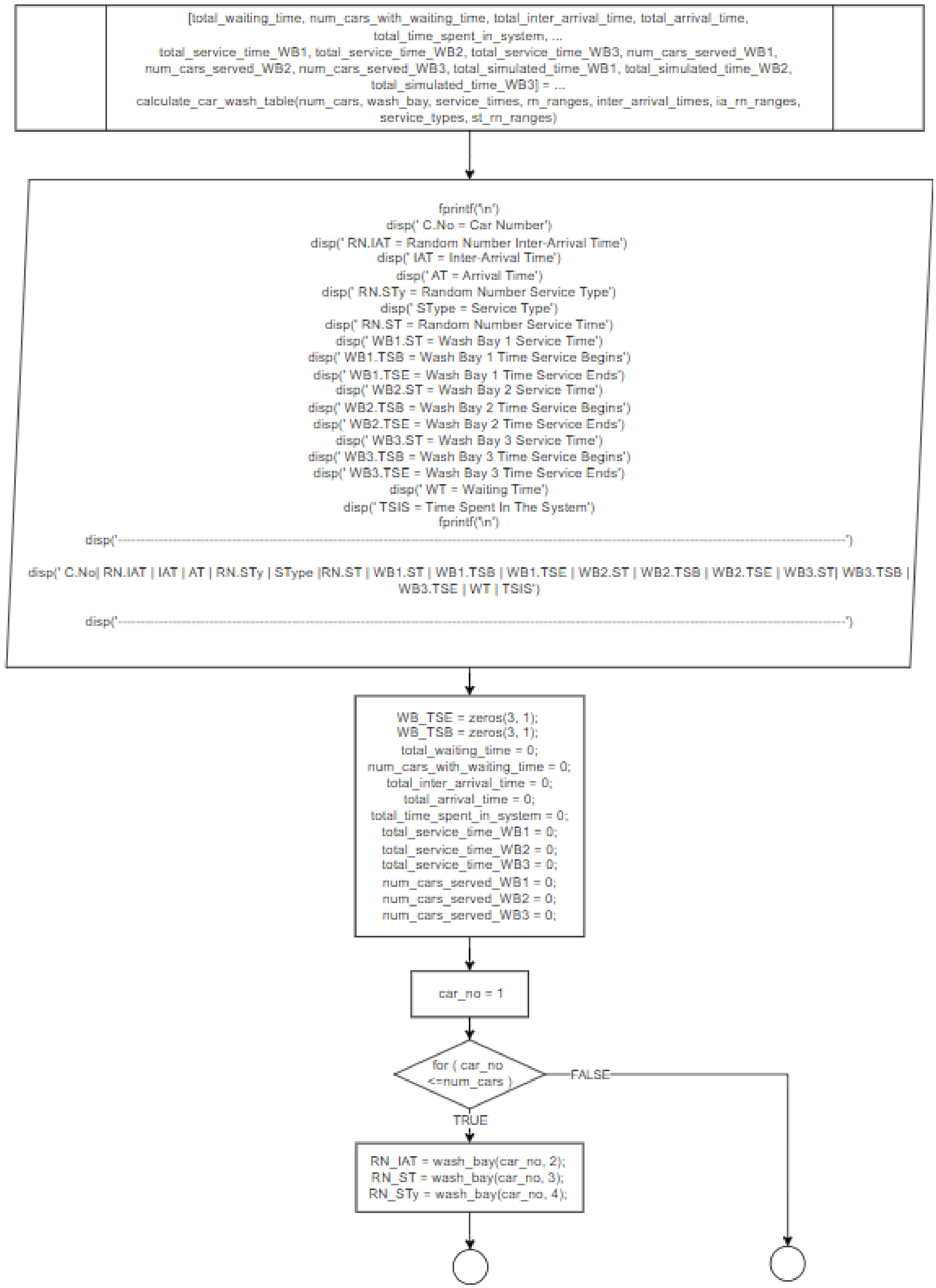
**calculate_averages:**
Computes and displays the average waiting time, inter-arrival time, arrival time, time spent in the system, and service time for each wash bay.
Calculates the probability that a car owner has to wait in the queue.

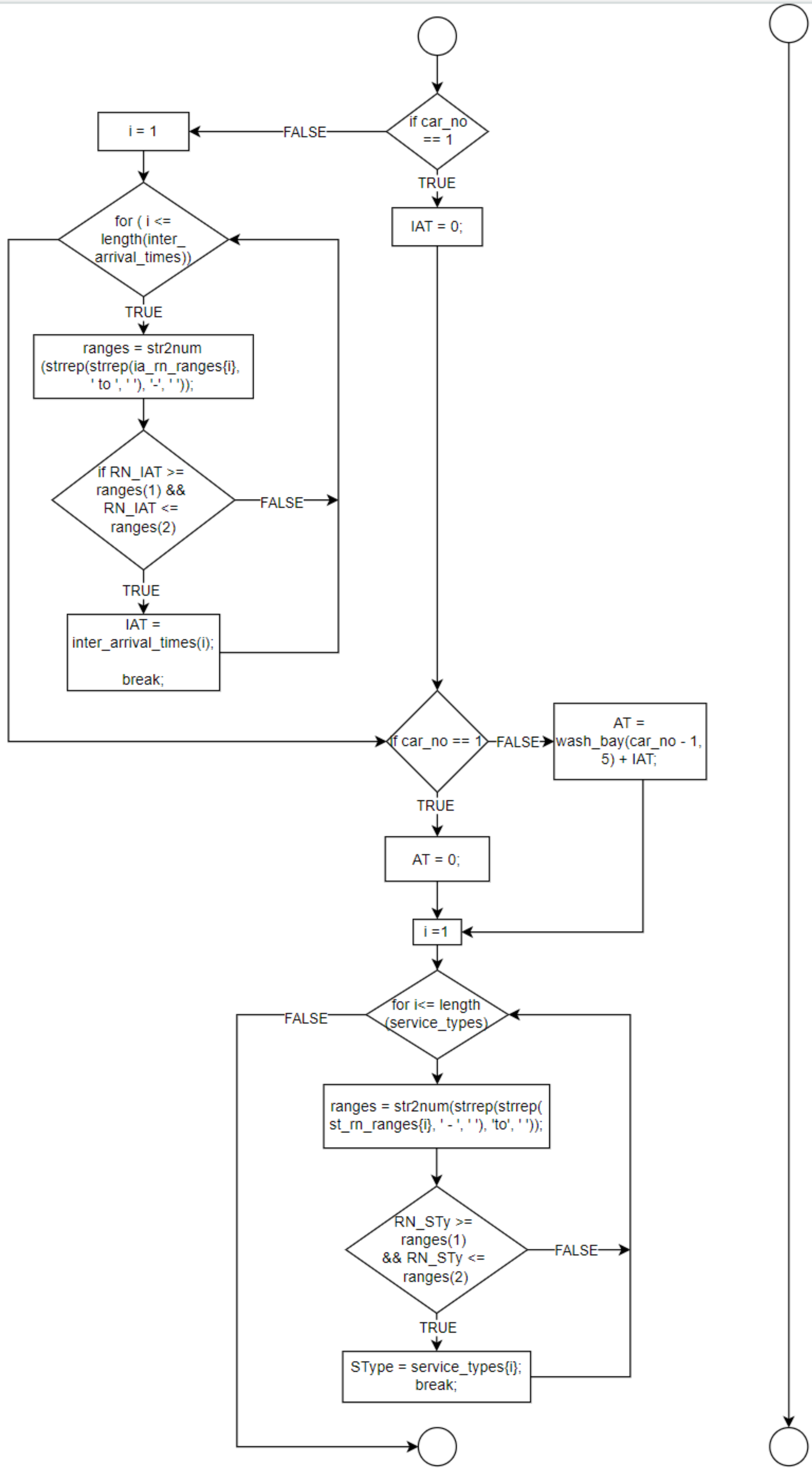
# FLOW-CHART

## Main.m

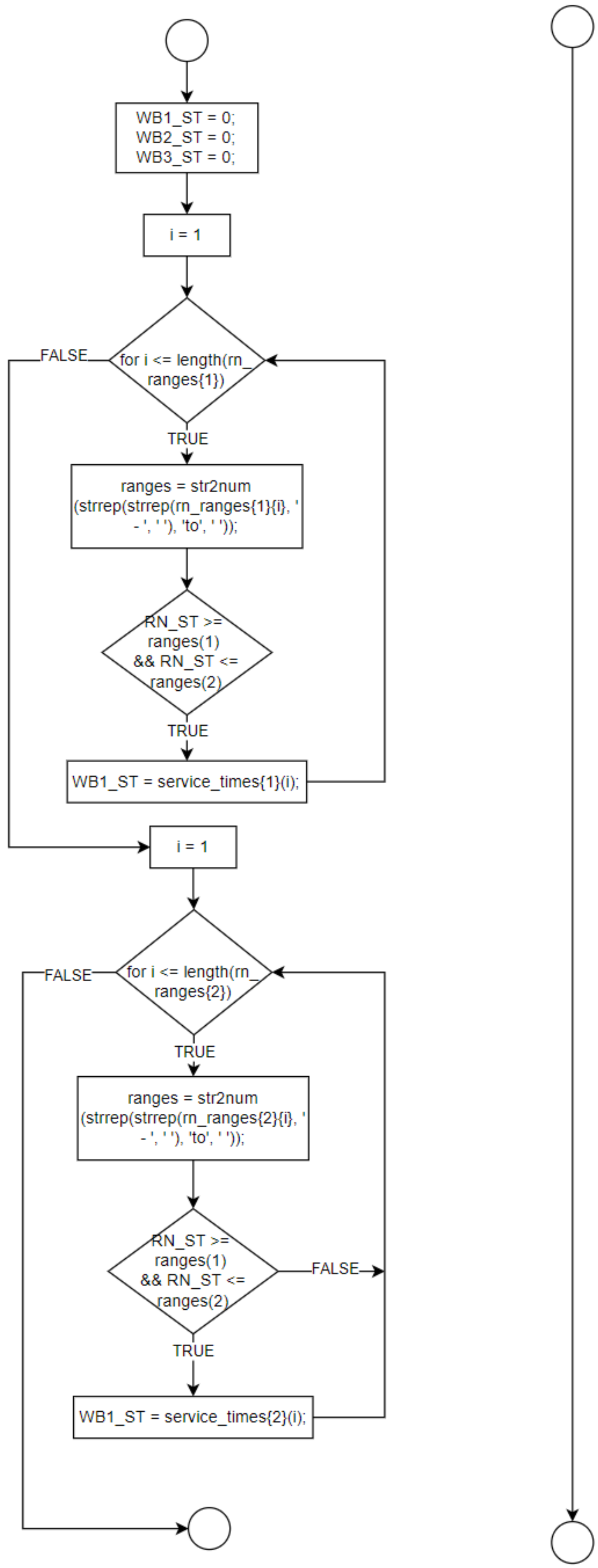


# calculate_car_wash_table.m

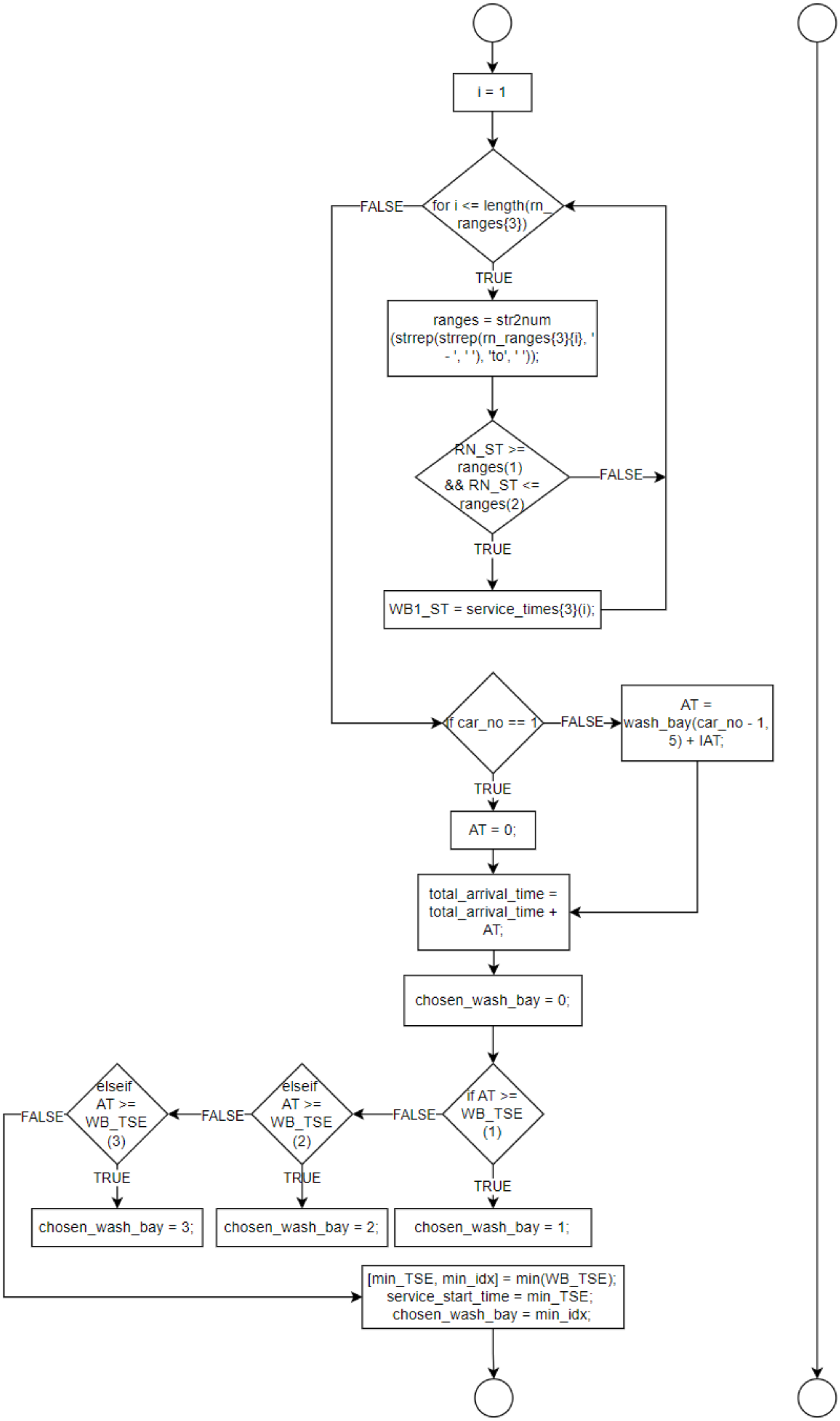




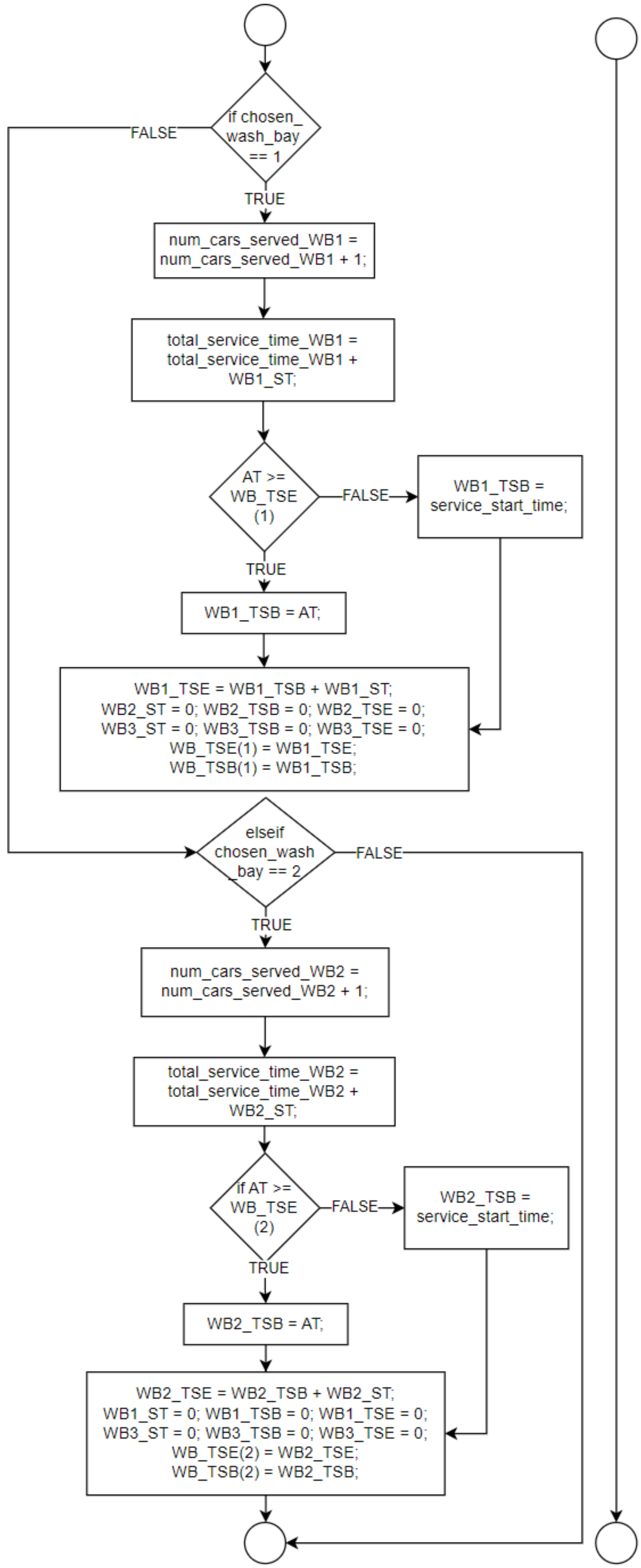
if car_no == 1
FALSE
TRUE
i = 1
IAT = 0;
for ( i <= length(inter_ arrival_times))
TRUE
ranges = str2num (strrep(strrep(ia_rn_ranges{i}, ' to ', ' '), '-', ' '));
if RN_IAT >= ranges(1) && RN_IAT <= ranges(2)
FALSE
TRUE
IAT = inter_arrival_times(i);
break;
if car_no == 1
FALSE
AT = wash_bay(car_no - 1, 5) + IAT;
TRUE
AT = 0;
i =1
for i<= length (service_types)
FALSE
ranges = str2num(strrep(strrep( st_rn_ranges{i}, ' - ', ' '), 'to', ' '));
RN_STy >= ranges(1) && RN_STy <= ranges(2)
FALSE
TRUE
SType = service_types{i}; break;

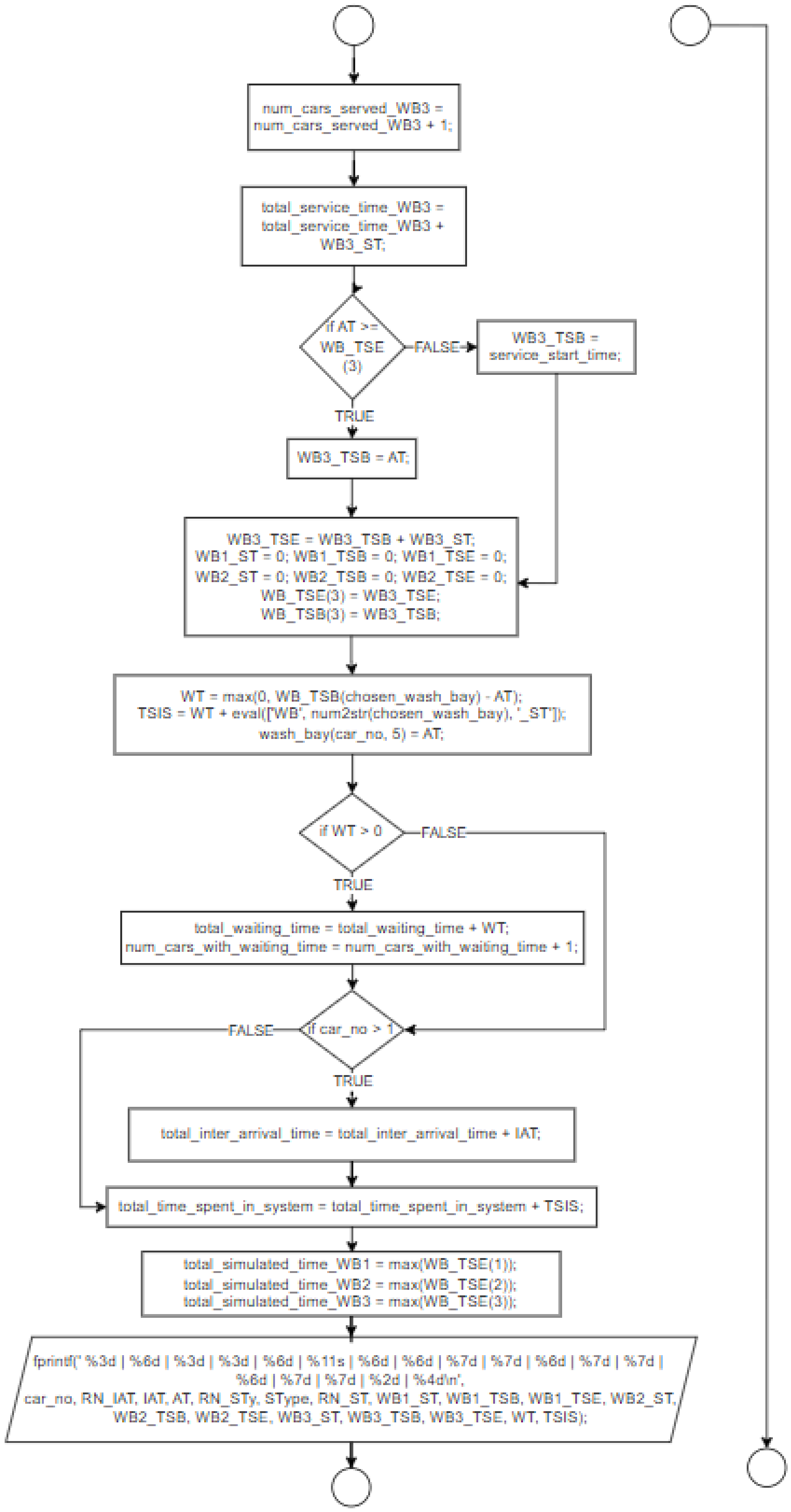


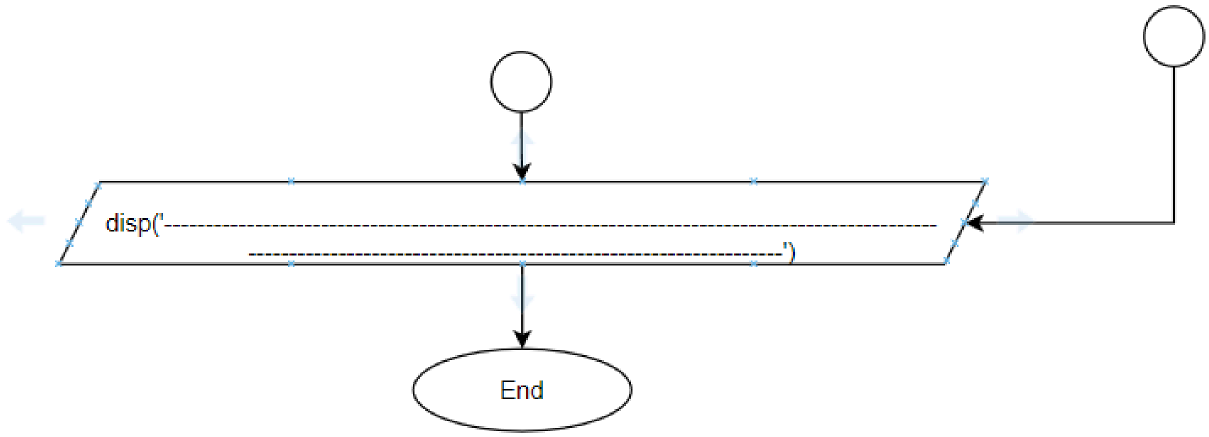
WB1_ST = 0;
WB2_ST = 0;
WB3_ST = 0;
i = 1
for i <= length(rn_ranges{1})
FALSE
TRUE
ranges = str2num (strrep(strrep(rn_ranges{1}{i}, ' - ', ' '), 'to', ' '));
RN_ST >= ranges(1) && RN_ST <= ranges(2)
TRUE
WB1_ST = service_times{1}(i);
i = 1
for i <= length(rn_ranges{2})
FALSE
TRUE
ranges = str2num (strrep(strrep(rn_ranges{2}{i}, ' - ', ' '), 'to', ' '));
RN_ST >= ranges(1) && RN_ST <= ranges(2)
FALSE
TRUE
WB1_ST = service_times{2}(i);



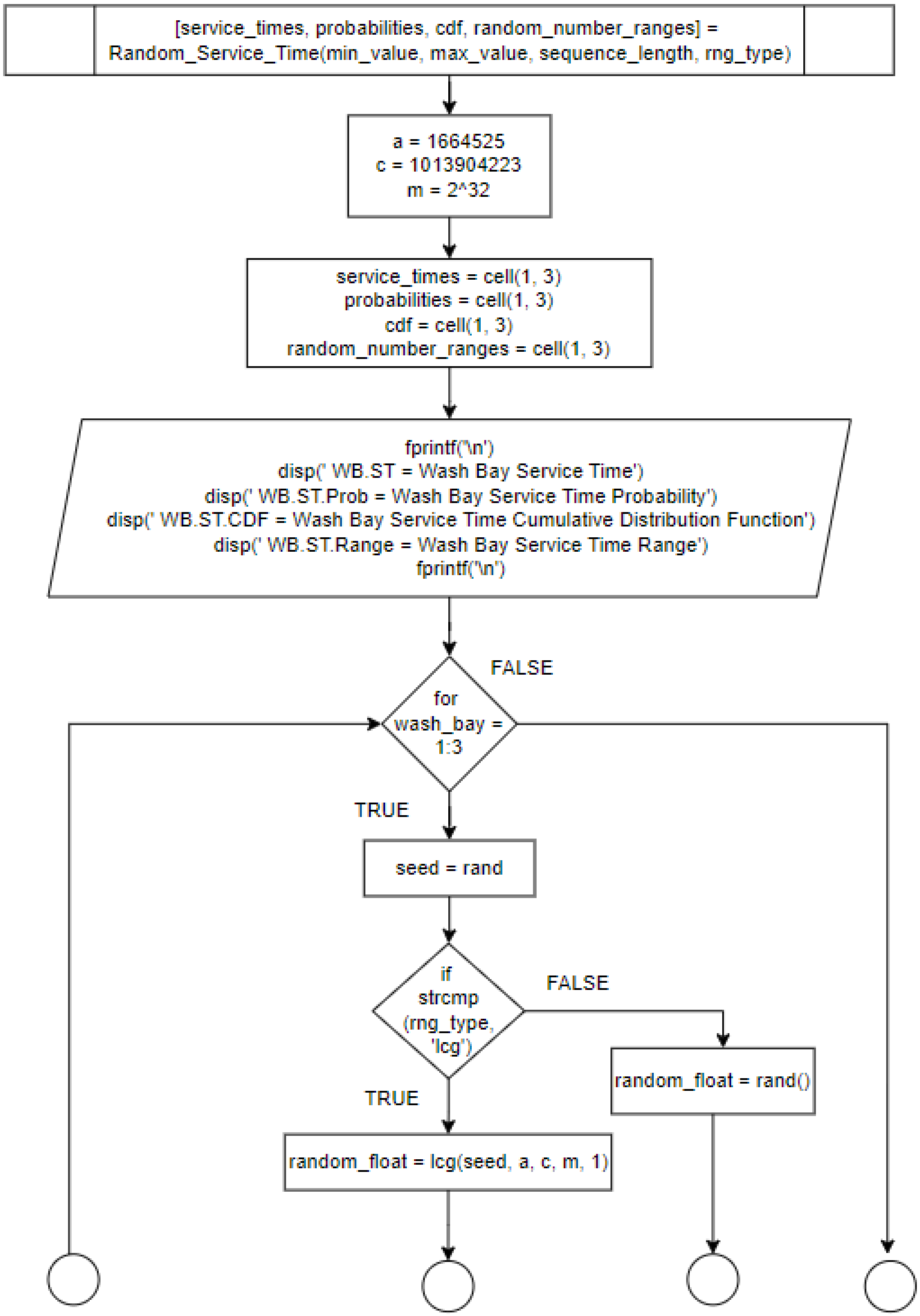
i = 1
for i <= length(rn_ranges{3})
FALSE
TRUE
ranges = str2num (strrep(strrep(rn_ranges{3}{i}, ' - ', ' '), 'to', ' '));
RN_ST >= ranges(1) && RN_ST <= ranges(2)
FALSE
TRUE
WB1_ST = service_times{3}(i);
if car_no == 1
FALSE
AT = wash_bay(car_no - 1, 5) + IAT;
TRUE
AT = 0;
total_arrival_time = total_arrival_time + AT;
chosen_wash_bay = 0;
if AT >= WB_TSE (1)
FALSE
elseif AT >= WB_TSE (2)
FALSE
elseif AT >= WB_TSE (3)
FALSE
TRUE
TRUE
TRUE
chosen_wash_bay = 3;
chosen_wash_bay = 2;
chosen_wash_bay = 1;
[min_TSE, min_idx] = min(WB_TSE);
service_start_time = min_TSE;
chosen_wash_bay = min_idx;

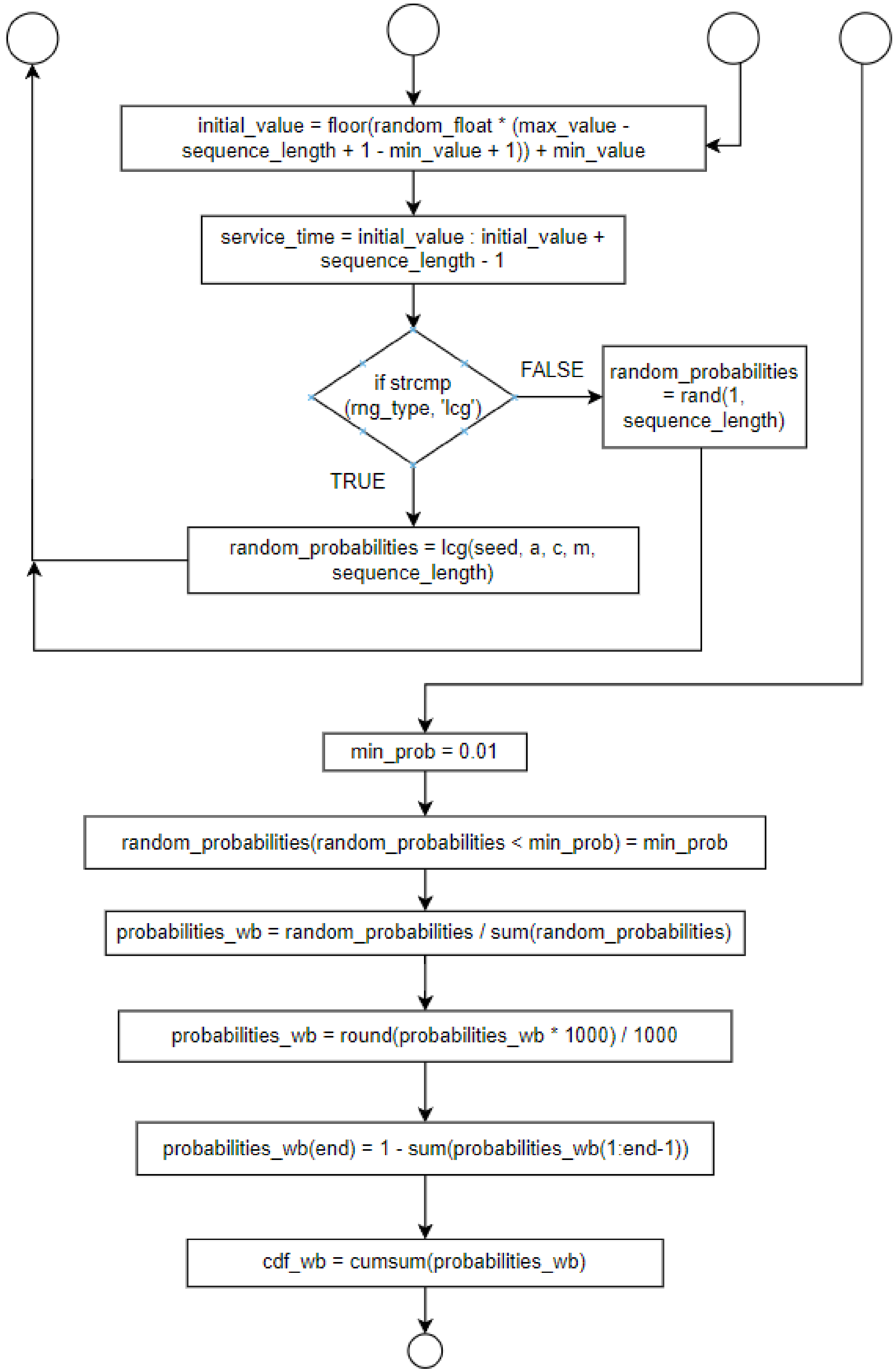

if chosen_ wash_bay == 1
FALSE
TRUE
num_cars_served_WB1 = num_cars_served_WB1 + 1;
total_service_time_WB1 = total_service_time_WB1 + WB1_ST;
AT >= WB_TSE (1)
FALSE
WB1_TSB = service_start_time;
TRUE
WB1_TSB = AT;
WB1_TSE = WB1_TSB + WB1_ST;
WB2_ST = 0; WB2_TSB = 0; WB2_TSE = 0;
WB3_ST = 0; WB3_TSB = 0; WB3_TSE = 0;
WB_TSE(1) = WB1_TSE;
WB_TSB(1) = WB1_TSB;
elseif chosen_wash _bay == 2
FALSE
TRUE
num_cars_served_WB2 = num_cars_served_WB2 + 1;
total_service_time_WB2 = total_service_time_WB2 + WB2_ST;
if AT >= WB_TSE (2)
FALSE
WB2_TSB = service_start_time;
TRUE
WB2_TSB = AT;
WB2_TSE = WB2_TSB + WB2_ST;
WB1_ST = 0; WB1_TSB = 0; WB1_TSE = 0;
WB3_ST = 0; WB3_TSB = 0; WB3_TSE = 0;
WB_TSE(2) = WB2_TSE;
WB_TSB(2) = WB2_TSB;


num_cars_served_WB3 = num_cars_served_WB3 + 1;
total_service_time_WB3 = total_service_time_WB3 + WB3_ST;
if AT >= WB_TSE (3)
FALSE
WB3_TSB = service_start_time;
TRUE
WB3_TSB = AT;
WB3_TSE = WB3_TSB + WB3_ST;
WB1_ST = 0; WB1_TSB = 0; WB1_TSE = 0;
WB2_ST = 0; WB2_TSB = 0; WB2_TSE = 0;
WB_TSE(3) = WB3_TSE;
WB_TSB(3) = WB3_TSB;
WT = max(0, WB_TSB(chosen_wash_bay) - AT);
TSIS = WT + eval(['WB', num2str(chosen_wash_bay), '_ST']);
wash_bay(car_no, 5) = AT;
if WT > 0
FALSE
TRUE
total_waiting_time = total_waiting_time + WT;
num_cars_with_waiting_time = num_cars_with_waiting_time + 1;
if car_no > 1
FALSE
TRUE
total_inter_arrival_time = total_inter_arrival_time + IAT;
total_time_spent_in_system = total_time_spent_in_system + TSIS;
total_simulated_time_WB1 = max(WB_TSE(1));
total_simulated_time_WB2 = max(WB_TSE(2));
total_simulated_time_WB3 = max(WB_TSE(3));
fprintf(' %3d | %6d | %3d | %3d | %6d | %11s | %6d | %6d | %7d | %7d | %6d | %7d | %7d | %6d | %7d | %7d | %2d | %4d\n',
car_no, RN_IAT, IAT, AT, RN_STy, SType, RN_ST, WB1_ST, WB1_TSB, WB1_TSE, WB2_ST, WB2_TSB, WB2_TSE, WB3_ST, WB3_TSB, WB3_TSE, WT, TSIS);

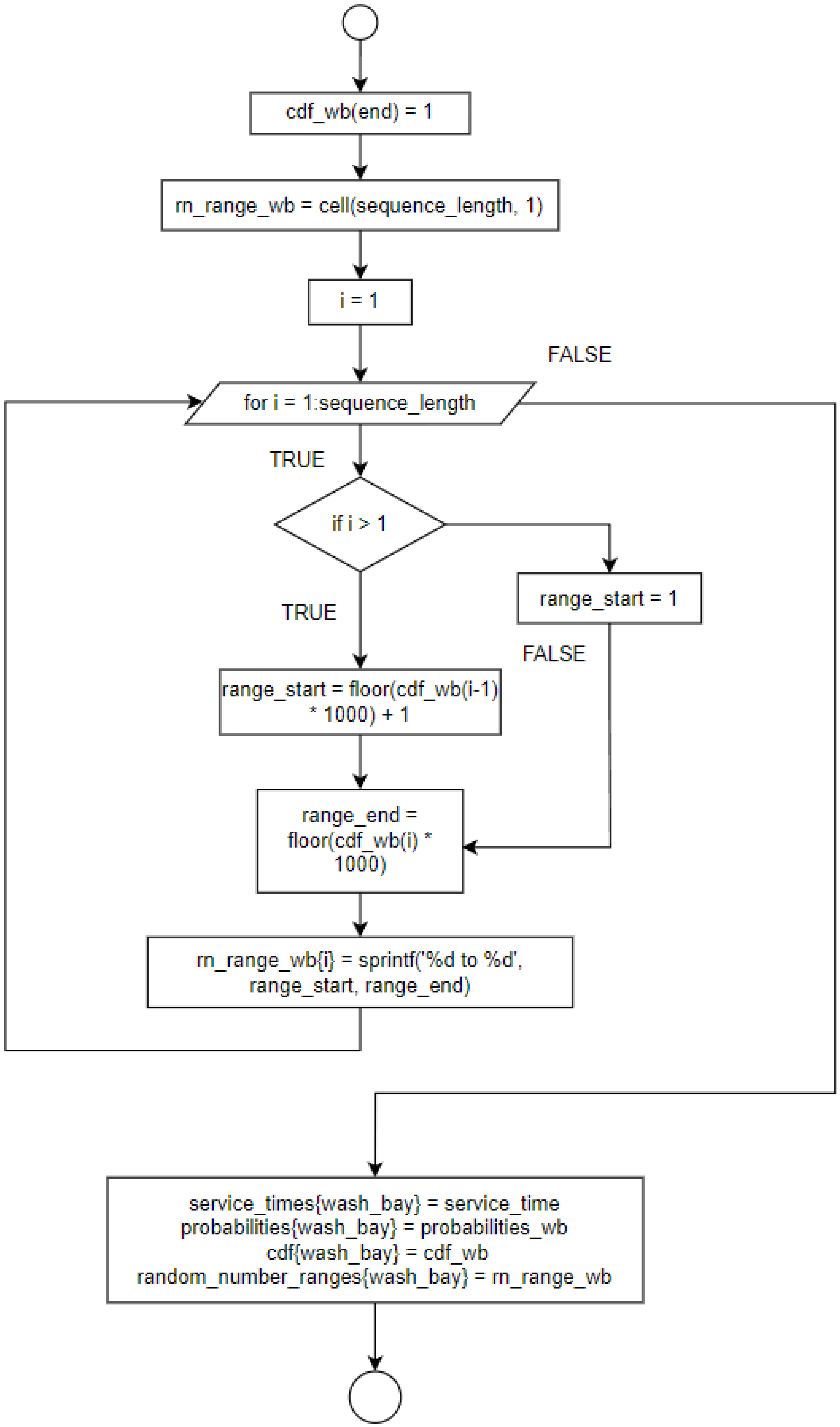

disp('-------------------------------------------------------------------------------------------------------------------------------------------------')
End

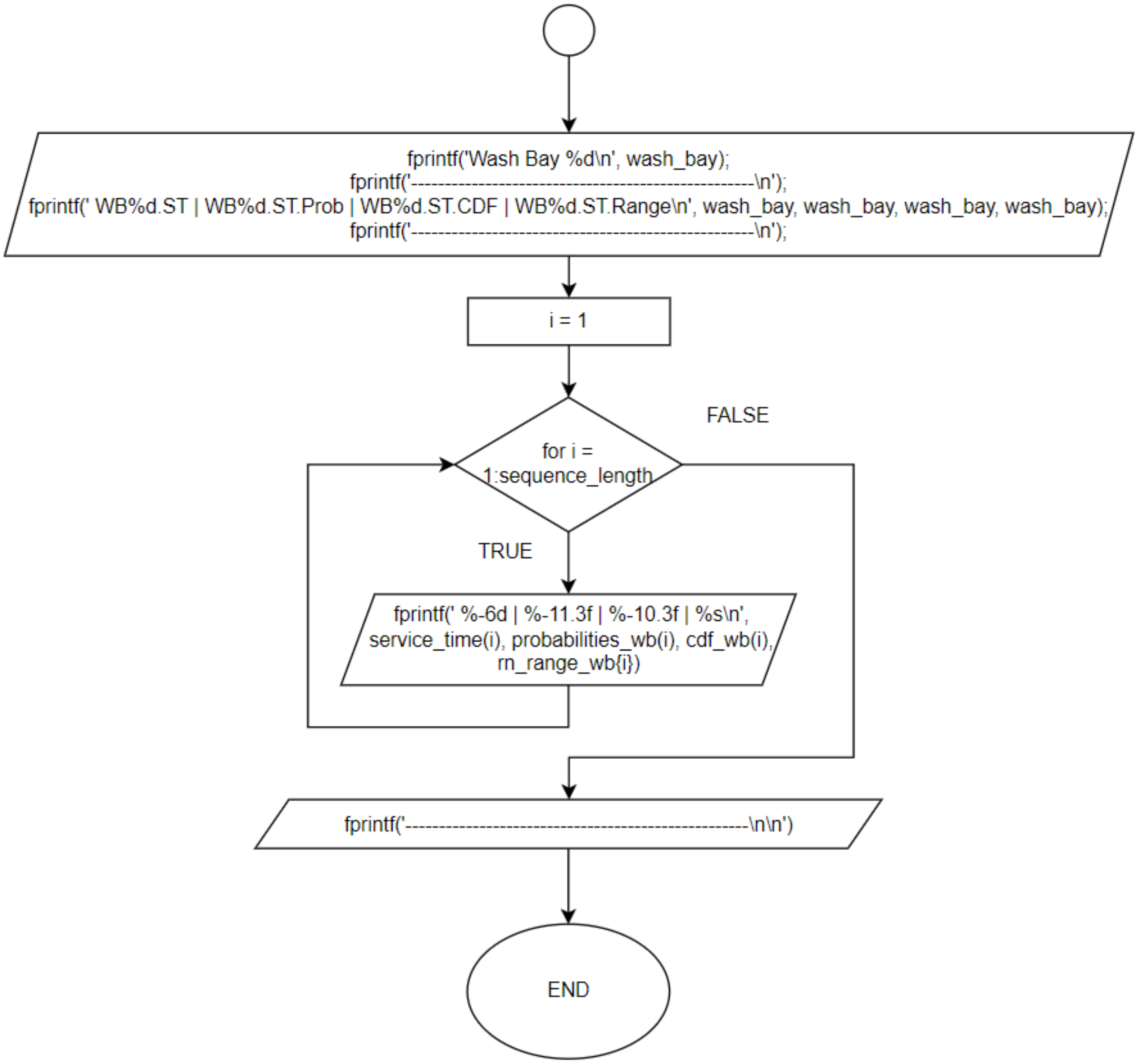
# Random_Service_Time.m



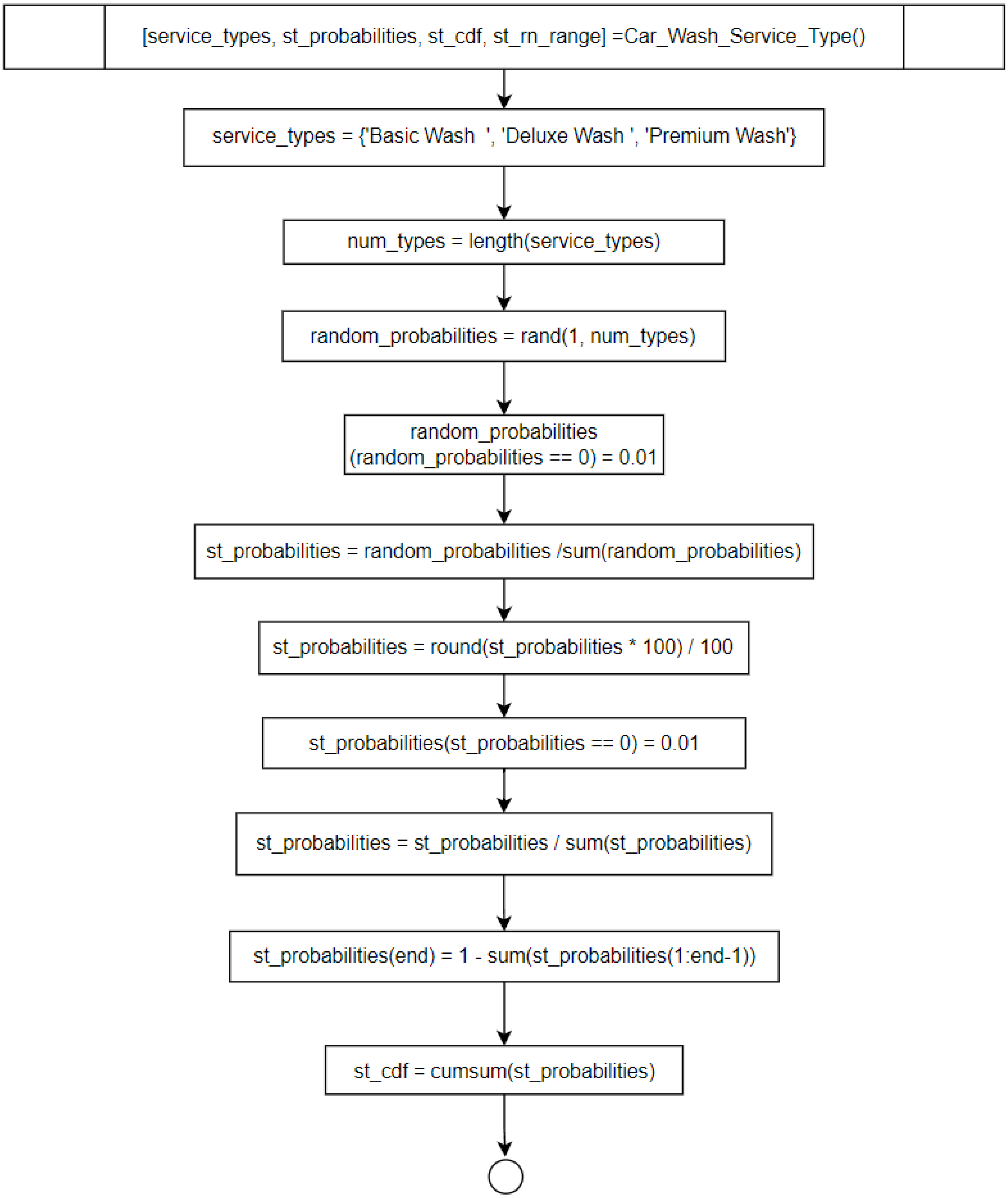


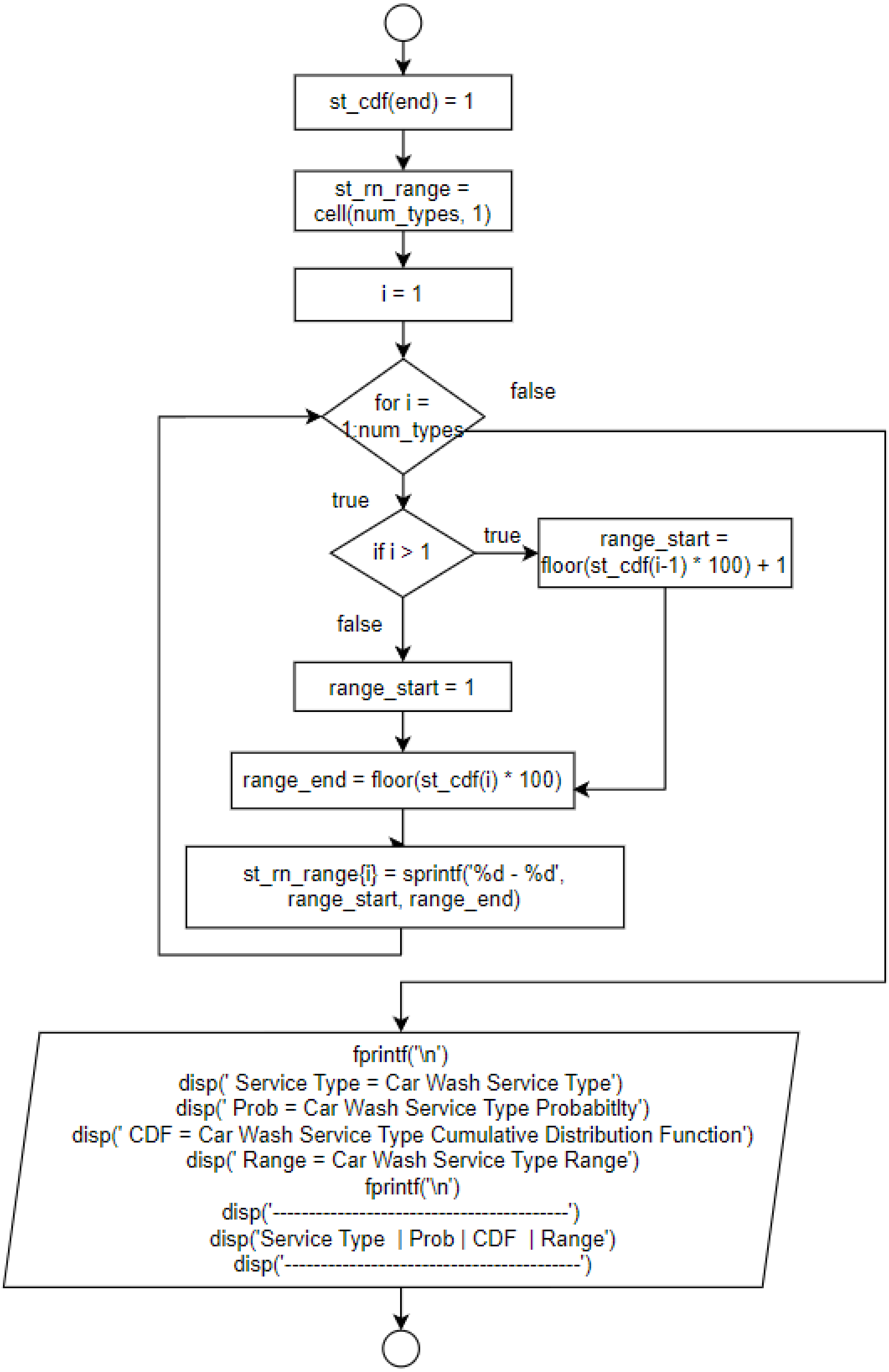
initial_value = floor(random_float * (max_value - sequence_length + 1 - min_value + 1)) + min_value
service_time = initial_value : initial_value + sequence_length - 1
if strcmp (rng_type, 'lcg')
FALSE
random_probabilities = rand(1, sequence_length)
TRUE
random_probabilities = lcg(seed, a, c, m, sequence_length)
min_prob = 0.01
random_probabilities(random_probabilities < min_prob) = min_prob
probabilities_wb = random_probabilities / sum(random_probabilities)
probabilities_wb = round(probabilities_wb * 1000) / 1000
probabilities_wb(end) = 1 - sum(probabilities_wb(1:end-1))
cdf_wb = cumsum(probabilities_wb)

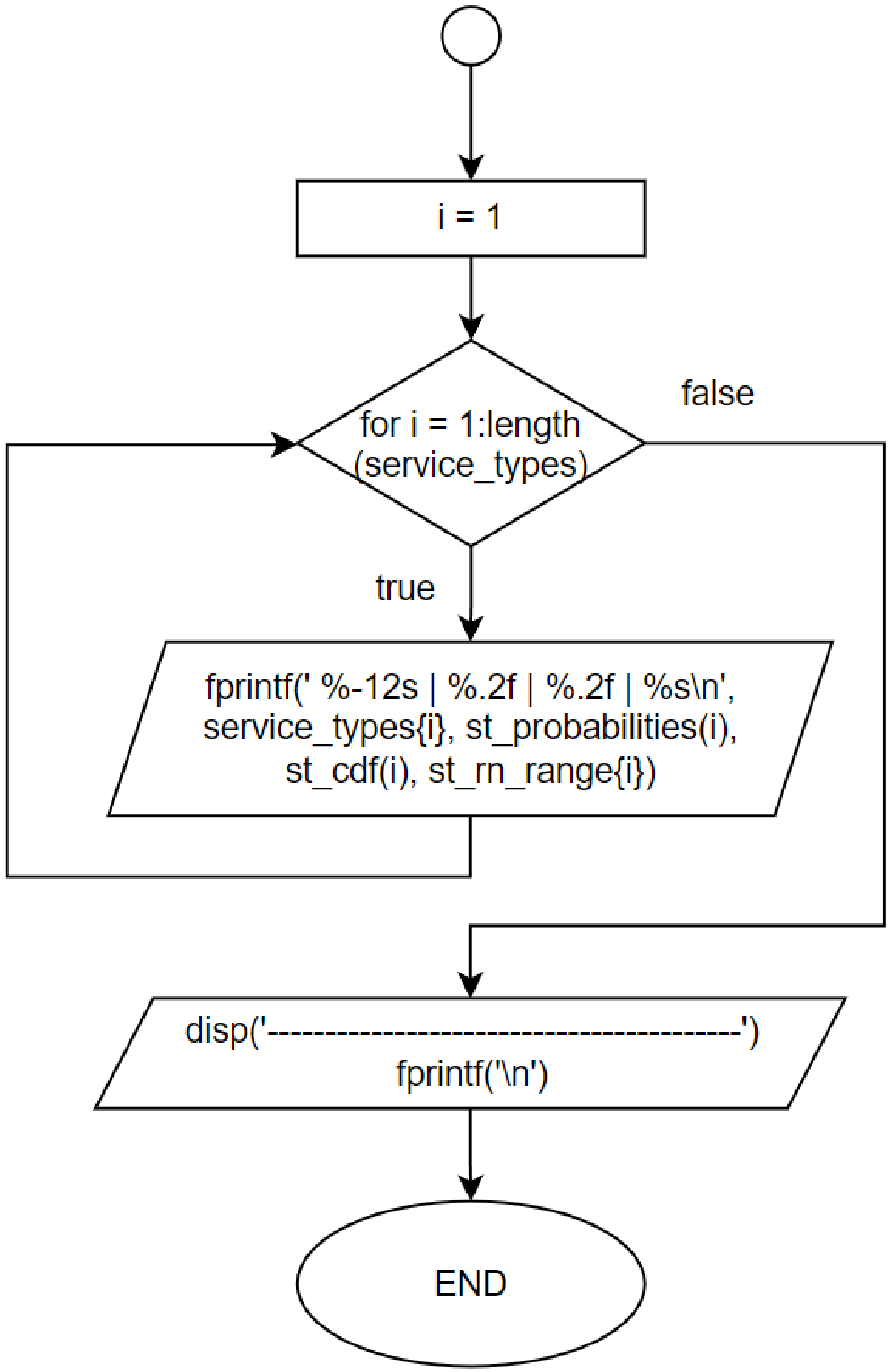

cdf_wb(end) = 1
rn_range_wb = cell(sequence_length, 1)
i = 1
FALSE
for i = 1:sequence_length
TRUE
if i > 1
range_start = 1
TRUE
FALSE
range_start = floor(cdf_wb(i-1) * 1000) + 1
range_end = floor(cdf_wb(i) * 1000)
rn_range_wb{i} = sprintf('%d to %d', range_start, range_end)
service_times{wash_bay} = service_time
probabilities{wash_bay} = probabilities_wb
cdf{wash_bay} = cdf_wb
random_number_ranges{wash_bay} = rn_range_wb


fprintf('Wash Bay %d\n', wash_bay);
fprintf('-------------------------------------------------\n');
fprintf(' WB%d.ST | WB%d.ST.Prob | WB%d.ST.CDF | WB%d.ST.Range\n', wash_bay, wash_bay, wash_bay, wash_bay);
fprintf('-------------------------------------------------\n');
i = 1
FALSE
for i = 1:sequence_length
TRUE
fprintf(' %-6d | %-11.3f | %-10.3f | %s\n', service_time(i), probabilities_wb(i), cdf_wb(i), rn_range_wb{i})
fprintf('-------------------------------------------------\n\n')
END

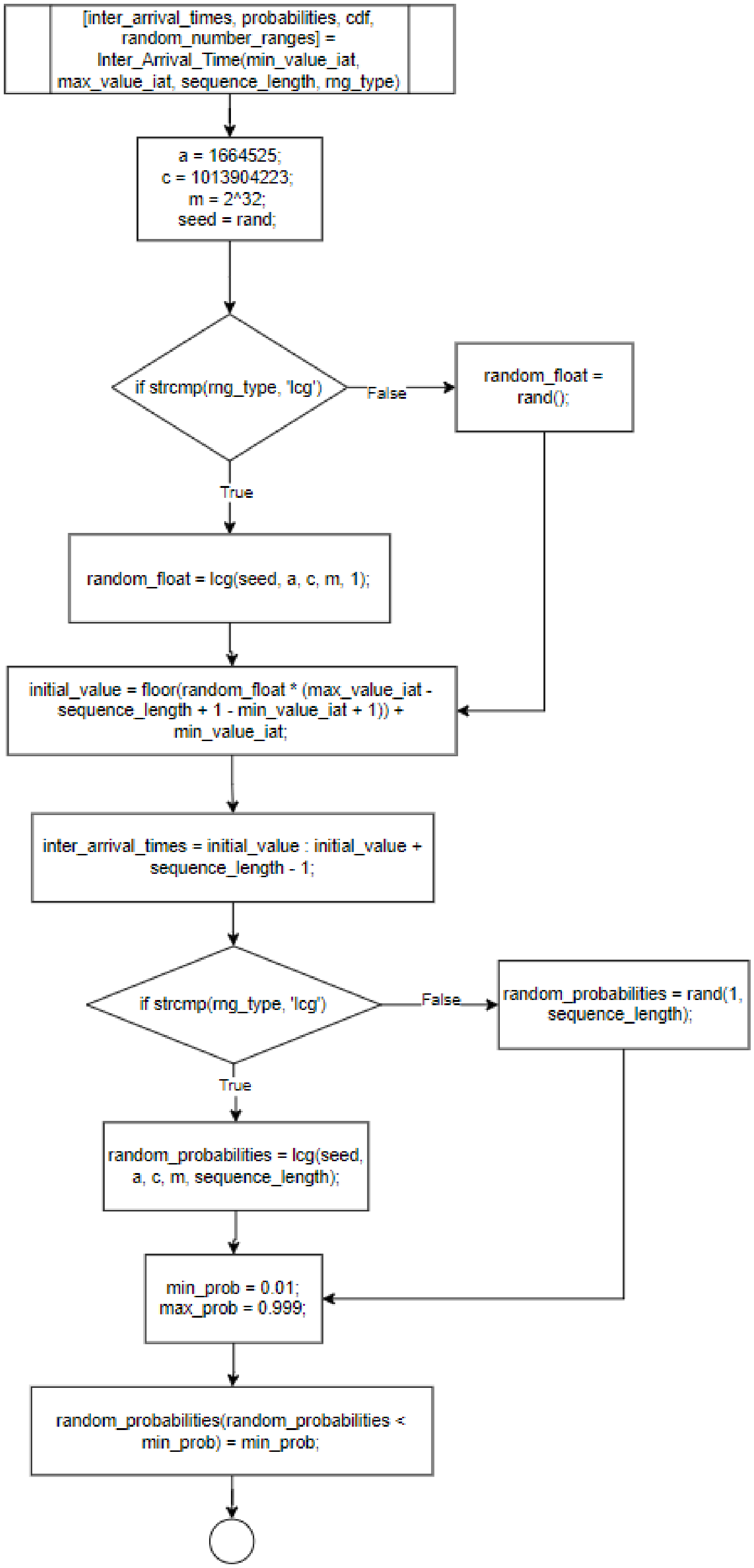
# Car_Wash_Service_Type.m

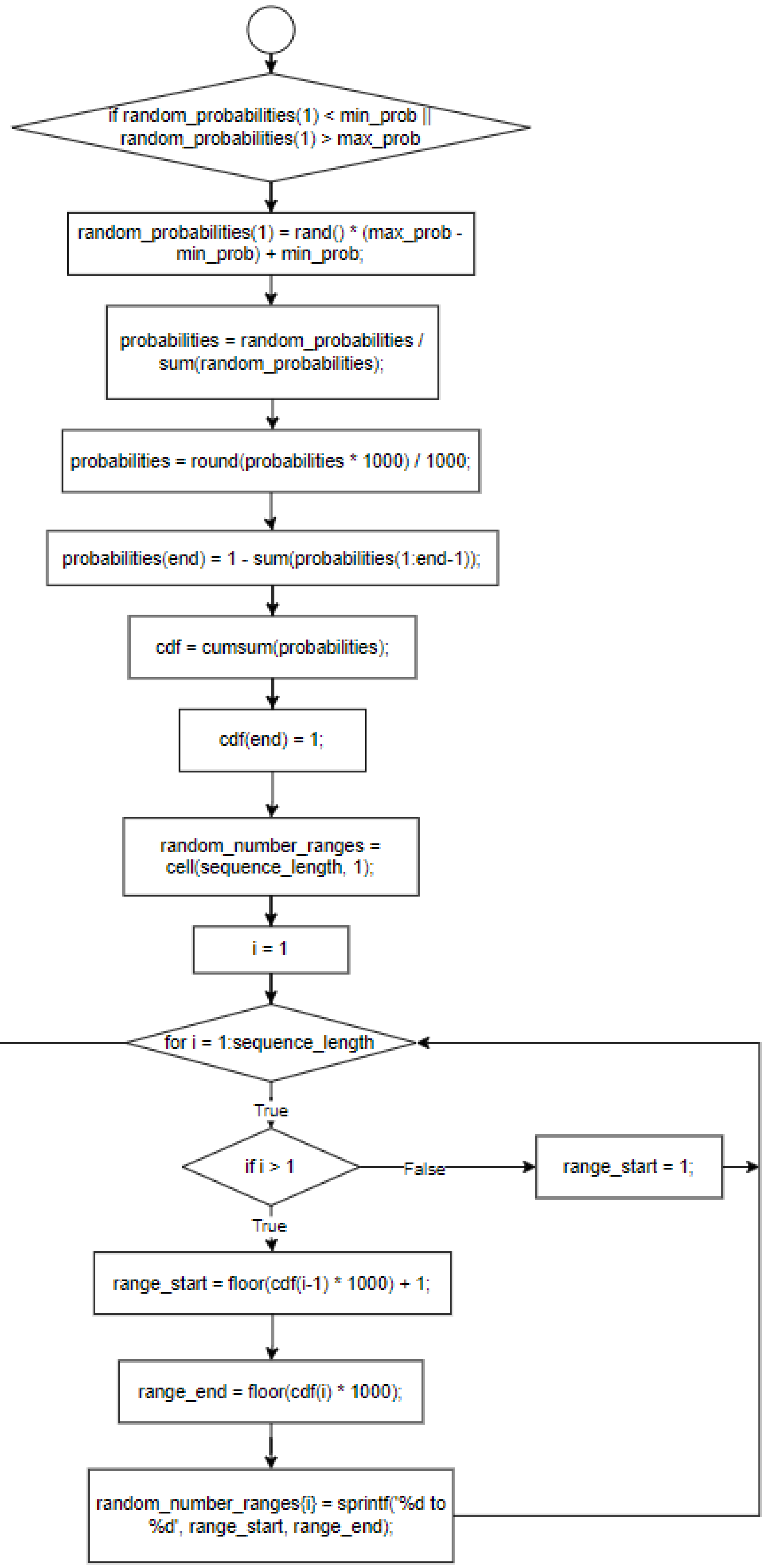



st_cdf(end) = 1
st_rn_range = cell(num_types, 1)
i = 1
for i = 1:num_types
false
true
if i > 1
true
range_start = floor(st_cdf(i-1) * 100) + 1
false
range_start = 1
range_end = floor(st_cdf(i) * 100)
st_rn_range{i} = sprintf('%d - %d', range_start, range_end)
fprintf('\n')
disp(' Service Type = Car Wash Service Type')
disp(' Prob = Car Wash Service Type Probabitlty')
disp(' CDF = Car Wash Service Type Cumulative Distribution Function')
disp(' Range = Car Wash Service Type Range')
fprintf('\n')
disp('----------------------------------------')
disp('Service Type | Prob | CDF | Range')
disp('----------------------------------------')

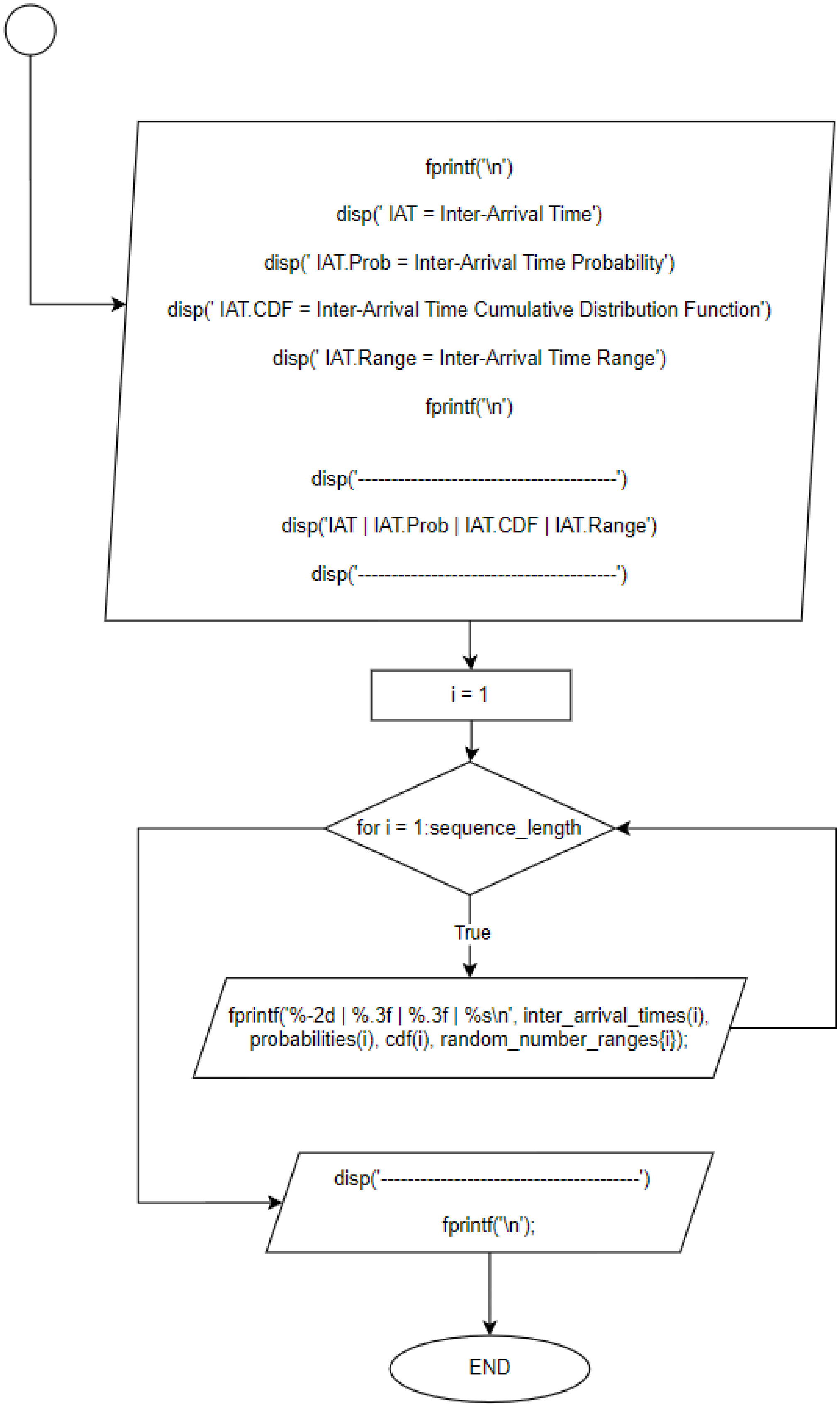

i = 1
for i = 1:length (service_types)
false
true
fprintf(' %-12s | %.2f | %.2f | %s\n', service_types{i}, st_probabilities(i), st_cdf(i), st_rn_range{i})
disp('---------------------------------------')
fprintf('\n')
END

# Inter_Arrival_Time.m

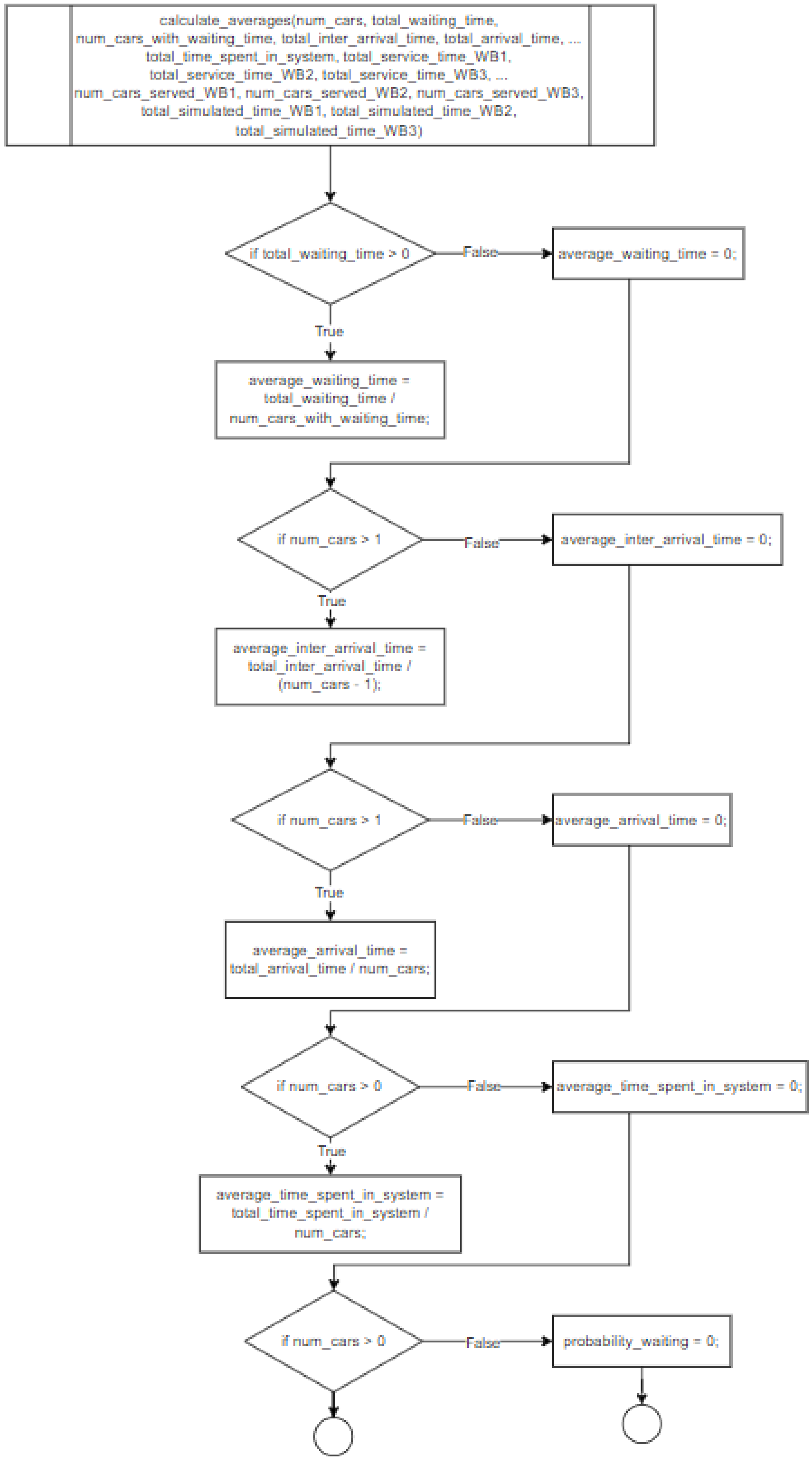


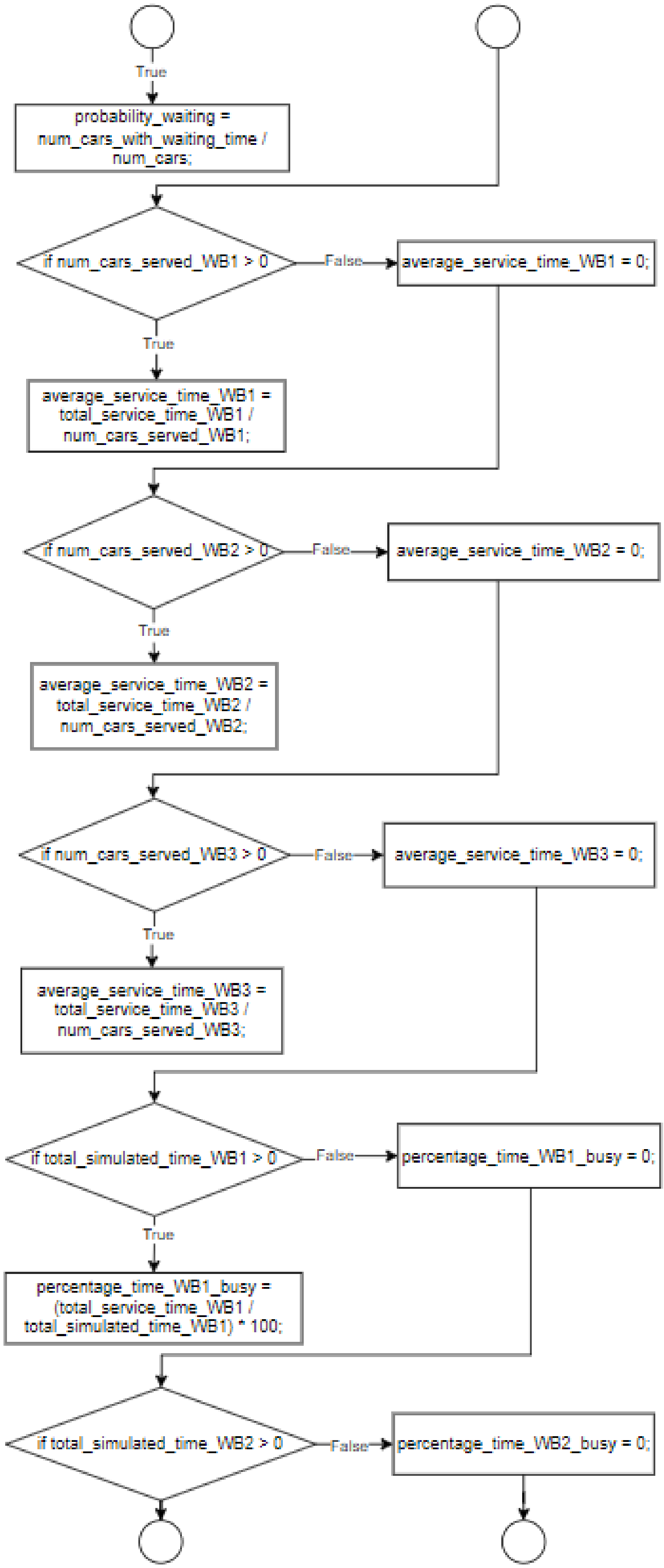


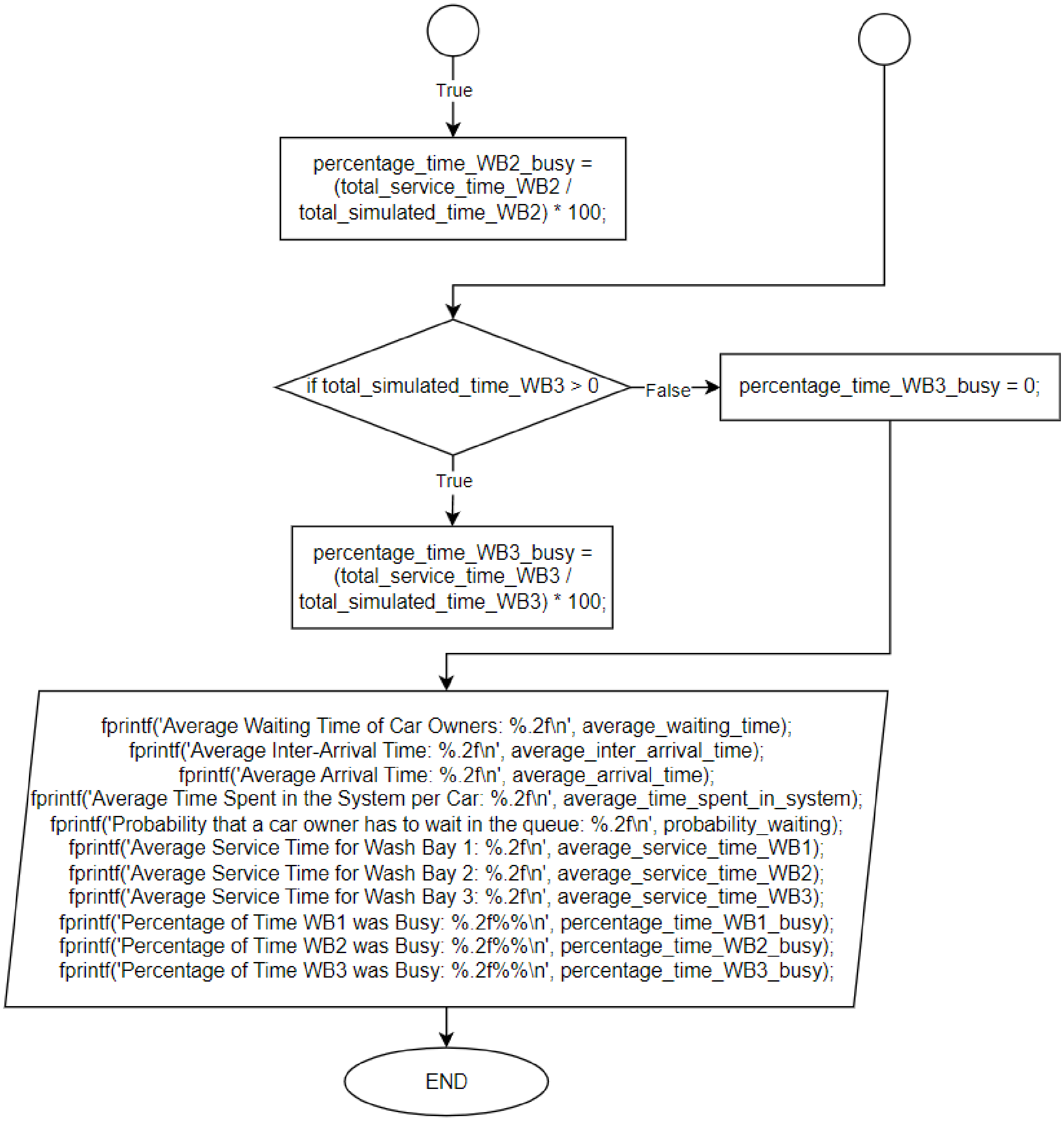
if random_probabilities(1) < min_prob || random_probabilities(1) > max_prob
random_probabilities(1) = rand() * (max_prob - min_prob) + min_prob;
probabilities = random_probabilities / sum(random_probabilities);
probabilities = round(probabilities * 1000) / 1000;
probabilities(end) = 1 - sum(probabilities(1:end-1));
cdf = cumsum(probabilities);
cdf(end) = 1;
random_number_ranges = cell(sequence_length, 1);
i = 1
for i = 1:sequence_length
True
if i > 1
False
range_start = 1;
True
range_start = floor(cdf(i-1) * 1000) + 1;
range_end = floor(cdf(i) * 1000);
random_number_ranges{i} = sprintf('%d to %d', range_start, range_end);


fprintf('\n')
disp(' IAT = Inter-Arrival Time')
disp(' IAT.Prob = Inter-Arrival Time Probability')
disp(' IAT.CDF = Inter-Arrival Time Cumulative Distribution Function')
disp(' IAT.Range = Inter-Arrival Time Range')
fprintf('\n')
disp('--------------------------------------------')
disp('IAT | IAT.Prob | IAT.CDF | IAT.Range')
disp('--------------------------------------------')
i = 1
for i = 1:sequence_length
True
fprintf('%-2d | %.3f | %.3f | %s\n', inter_arrival_times(i), probabilities(i), cdf(i), random_number_ranges{i});
disp('--------------------------------------------')
fprintf('\n');
END

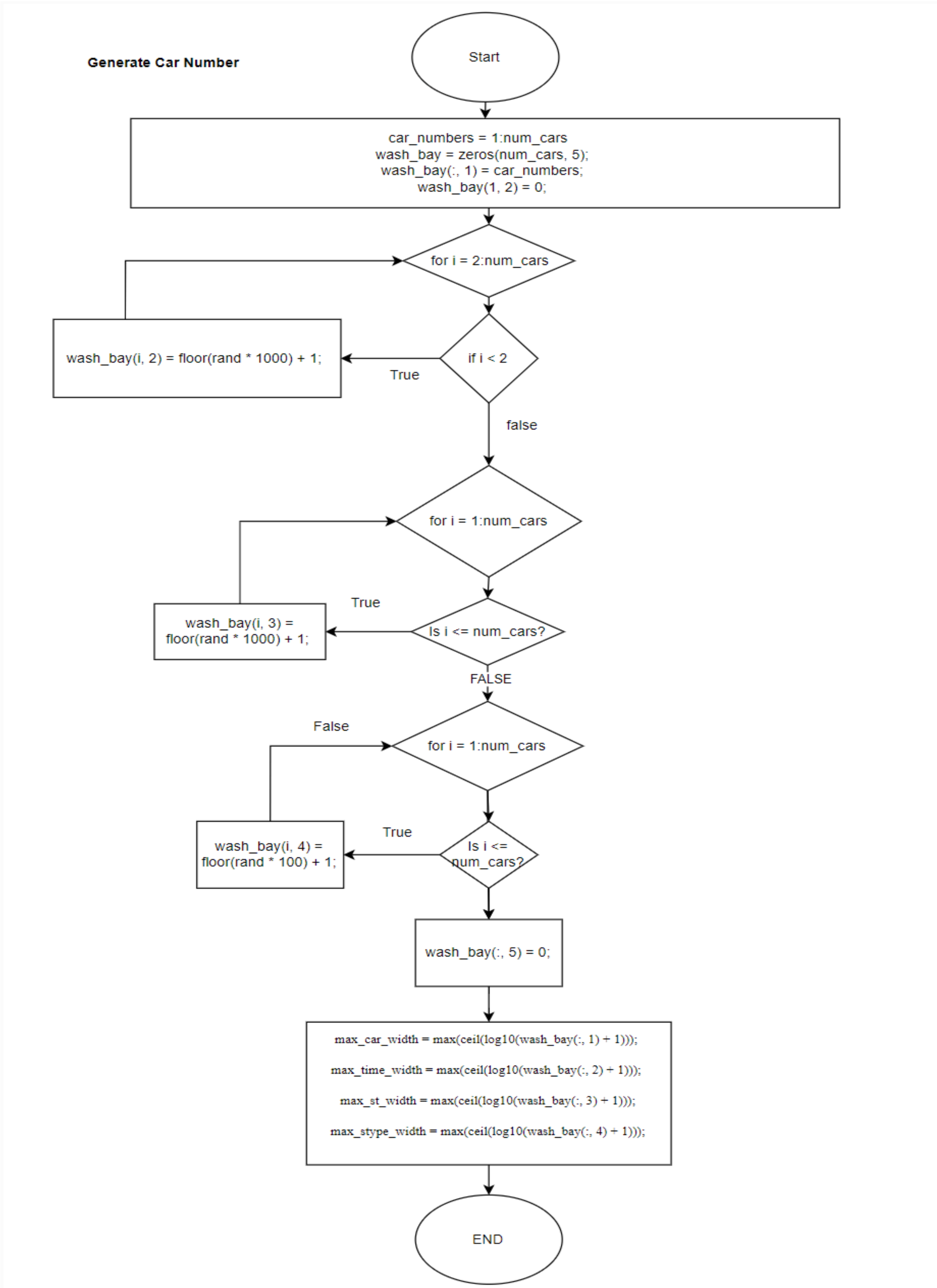
# calculate_averages.m



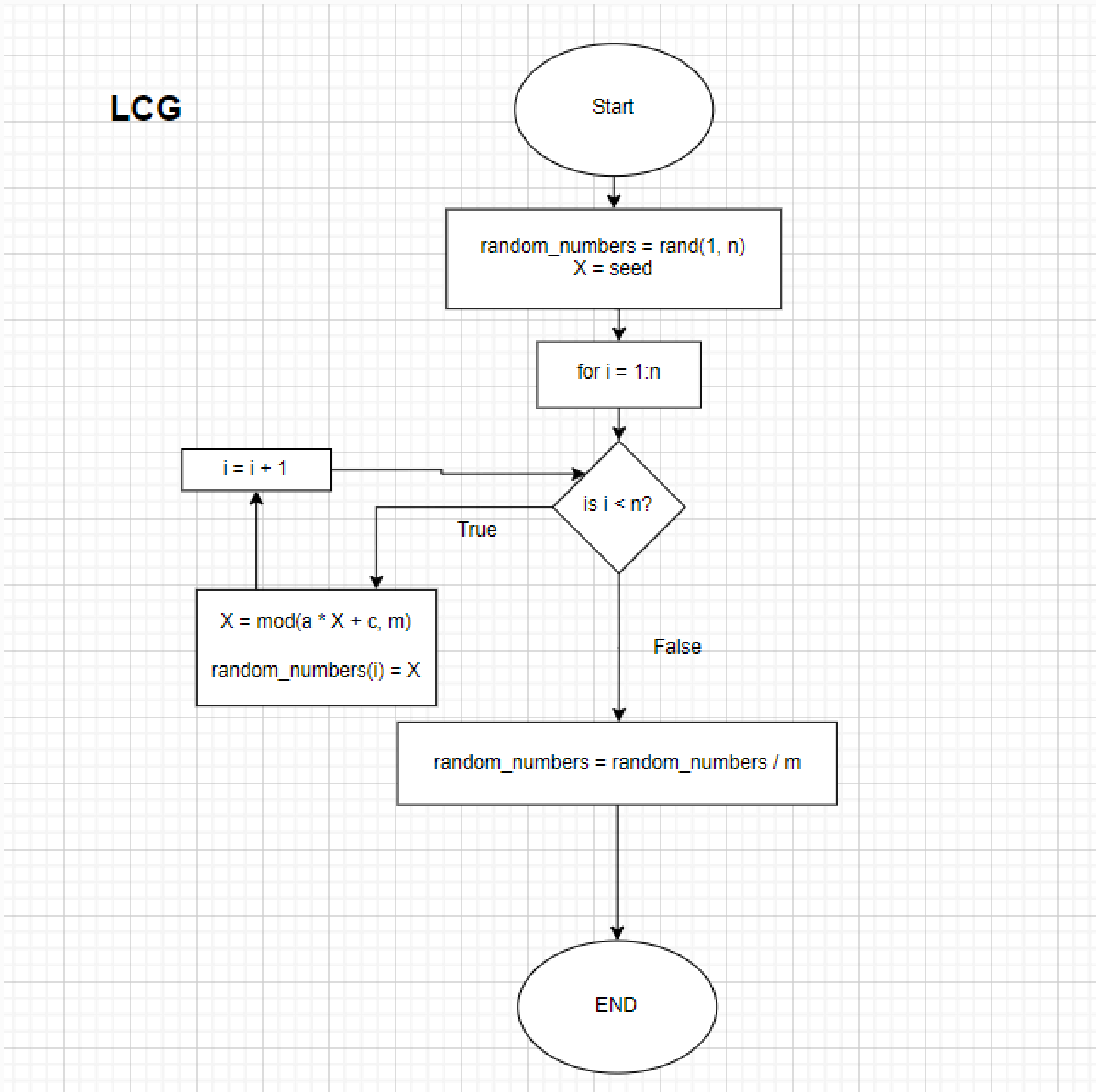

True
probability_waiting = num_cars_with_waiting_time / num_cars;
if num_cars_served_WB1 > 0
False
average_service_time_WB1 = 0;
True
average_service_time_WB1 = total_service_time_WB1 / num_cars_served_WB1;
if num_cars_served_WB2 > 0
False
average_service_time_WB2 = 0;
True
average_service_time_WB2 = total_service_time_WB2 / num_cars_served_WB2;
if num_cars_served_WB3 > 0
False
average_service_time_WB3 = 0;
True
average_service_time_WB3 = total_service_time_WB3 / num_cars_served_WB3;
if total_simulated_time_WB1 > 0
False
percentage_time_WB1_busy = 0;
True
percentage_time_WB1_busy = (total_service_time_WB1 / total_simulated_time_WB1) * 100;
if total_simulated_time_WB2 > 0
False
percentage_time_WB2_busy = 0;


True
percentage_time_WB2_busy = (total_service_time_WB2 / total_simulated_time_WB2) * 100;
if total_simulated_time_WB3 > 0
False
percentage_time_WB3_busy = 0;
True
percentage_time_WB3_busy = (total_service_time_WB3 / total_simulated_time_WB3) * 100;
fprintf('Average Waiting Time of Car Owners: %.2f\n', average_waiting_time);
fprintf('Average Inter-Arrival Time: %.2f\n', average_inter_arrival_time);
fprintf('Average Arrival Time: %.2f\n', average_arrival_time);
fprintf('Average Time Spent in the System per Car: %.2f\n', average_time_spent_in_system);
fprintf('Probability that a car owner has to wait in the queue: %.2f\n', probability_waiting);
fprintf('Average Service Time for Wash Bay 1: %.2f\n', average_service_time_WB1);
fprintf('Average Service Time for Wash Bay 2: %.2f\n', average_service_time_WB2);
fprintf('Average Service Time for Wash Bay 3: %.2f\n', average_service_time_WB3);
fprintf('Percentage of Time WB1 was Busy: %.2f%%\n', percentage_time_WB1_busy);
fprintf('Percentage of Time WB2 was Busy: %.2f%%\n', percentage_time_WB2_busy);
fprintf('Percentage of Time WB3 was Busy: %.2f%%\n', percentage_time_WB3_busy);
END

# Generate_Car_Numbers.m



## LCG.m



LCG
Start
random_numbers = rand(1, n)
X = seed
for i = 1:n
i = i + 1
is i < n?
True
X = mod(a * X + c, m)
random_numbers(i) = X
False
random_numbers = random_numbers / m
END

# Explanation for implementation important source codes

## Main.m

This script simulates a car wash system where the user chooses between 'rand' or 'lcg' for random number generation. It generates random service times, inter-arrival times, and service types based on specified ranges. After inputting the number of cars to simulate, the script calculates statistics like waiting times and service durations for each wash bay using the calculate_car_wash_table function. Finally, calculate_averages computes and displays average values and probabilities related to the car wash process, with error handling for invalid input types.

## calculate_car_wash_table.m

```
% Initialize variables for tracking wash bay service times
WB_TSE = zeros(3, 1);  % Array to store the end times of the three wash bays
WB_TSB = zeros(3, 1);  % Array to store the start times of the three wash bays
```

This code initializes two arrays, WB_TSE and WB_TSB, each with three elements set to zero. These arrays are used to store the start (WB_TSB) and end times (WB_TSE) of service for each of the three wash bays.

```
% Retrieve car data
RN_IAT = wash_bay(car_no, 2);  % Random Number Inter-Arrival Time
RN_ST = wash_bay(car_no, 3);   % Random Number Service Time
RN_STy = wash_bay(car_no, 4);  % Random Number Service Type
```

This 3 line of code extracts the random numbers associated with a specific car (car_no) from the wash_bay matrix. The variables RN_IAT, RN_ST, and RN_STy are assigned the random numbers for the car's inter-arrival time, service time, and service type, respectively, from columns 2, 3, and 4 of the wash_bay matrix.

For example, if car_no is 1 and the wash_bay matrix has the values [1, 500, 200, 50] in its first row, then RN_IAT will be 500, RN_ST will be 200, and RN_STy will be 50, corresponding to the random numbers for inter-arrival time, service time, and service type, respectively.

```
for i = 1:length(inter_arrival_times)
    % Parse the range string and convert to numbers
    ranges = str2num(strrep(strrep(ia_rn_ranges{i}, ' to ', ' '), '-', ' '));
    if RN_IAT >= ranges(1) && RN_IAT <= ranges(2)
        IAT = inter_arrival_times(i);
        break;
    end
end
```

for i = 1:length(inter_arrival_times)

The above line of code means loop iterates through each element in the inter_arrival_times array.

ranges = str2num(strrep(strrep(ia_rn_ranges{i}, ' to ', ' '), '-', ' '));
The above line of code processes the ia_rn_ranges{i} string to convert it into numerical values.
Replacing the text " to " with a space.
Replacing any hyphens '-' with a space.
Using str2num to convert the resulting string into an array of numbers.
Strrep mean replace occurrences of a substring within a string with another substring
For example, if ia_rn_ranges{i} is "100 to 200", this line would convert it to [100 200].

if RN_IAT >= ranges(1) && RN_IAT <= ranges(2)
This condition checks if the RN_IAT falls within the current range (ranges(1) to ranges(2)).

IAT = inter_arrival_times(i);
break;
If the RN_IAT falls within the current range, it assigns the corresponding inter_arrival_times(i) to IAT and breaks the loop, since the correct inter-arrival time has been found.

```
% Calculate Arrival Time (AT)
if car_no == 1
    AT = 0;  % First car arrival time is 0
else
    AT = wash_bay(car_no - 1, 5) + IAT;  % Arrival time is cumulative
end
```

For the first car (car_no == 1), the arrival time is set to 0 because there are no previous cars to consider.
For all subsequent cars, the arrival time is determined by adding the inter-arrival time (IAT) to the arrival time of the previous car (wash_bay(car_no - 1, 5)). This ensures that the arrival times are cumulative, reflecting the sequence in which cars arrive at the wash bay based on the generated inter-arrival times.

```matlab
% Check the end times of all wash bays to find the appropriate one
if AT >= WB_TSE(1)
    chosen_wash_bay = 1;
elseif AT >= WB_TSE(2)
    chosen_wash_bay = 2;
elseif AT >= WB_TSE(3)
    chosen_wash_bay = 3;
else
    % All bays are busy, choose the soonest available bay
    [min_TSE, min_idx] = min(WB_TSE);
    service_start_time = min_TSE; % Use min_TSE as the service start time
    chosen_wash_bay = min_idx;
end
```

This code segment determines which wash bay (1, 2, or 3) a car should be assigned to based on its arrival time (AT). It checks if any wash bays are available at the car's arrival time and, if all are busy, assigns the car to the wash bay that will be available the soonest.

if AT >= WB_TSE(1) chosen_wash_bay = 1;
elseif AT >= WB_TSE(2) chosen_wash_bay = 2;
elseif AT >= WB_TSE(3) chosen_wash_bay = 3;
The code checks if the car's arrival time (AT) is greater than or equal to the end time of service (WB_TSE) for each wash bay.

[min_TSE, min_idx] = min(WB_TSE);
service_start_time = min_TSE;
chosen_wash_bay = min_idx;
If AT is less than the end times of service for all wash bays (i.e., all wash bays are currently busy), the code finds the wash bay that will be available the soonest.
[min_TSE, min_idx] = min(WB_TSE); finds the minimum end time of service (min_TSE) and the corresponding index (min_idx), indicating which wash bay will be available first.
It sets service_start_time to min_TSE, indicating when the service can start for the current car.
chosen_wash_bay is set to min_idx, indicating the wash bay that will be available first.

```
% Assign the car to the chosen wash bay and reset other bays' data
if chosen_wash_bay == 1
    num_cars_served_WB1 = num_cars_served_WB1 + 1;
    % Update total service time for Wash Bay 1
    total_service_time_WB1 = total_service_time_WB1 + WB1_ST;
    if AT >= WB_TSE(1)
        WB1_TSB = AT;
    else
        WB1_TSB = service_start_time; % Use the service start time
    end
    WB1_TSE = WB1_TSB + WB1_ST;
    WB2_ST = 0; WB2_TSB = 0; WB2_TSE = 0;
    WB3_ST = 0; WB3_TSB = 0; WB3_TSE = 0;
    WB_TSE(1) = WB1_TSE;
    WB_TSB(1) = WB1_TSB;
```

If wash bay 1 is chosen (chosen_wash_bay == 1), it increments the count of cars served by wash bay 1 (num_cars_served_WB1). It then updates the total service time for wash bay 1 (total_service_time_WB1). The service start time (WB1_TSB) is set to the car's arrival time (AT) if the bay is available, otherwise, it uses the calculated service_start_time. The service end time (WB1_TSE) is then calculated by adding the service time (WB1_ST) to the start time. The service times and end times for wash bays 2 and 3 are reset to zero, and the overall end time array (WB_TSE) is updated with the new end time for wash bay 1.

```
% Calculate Waiting Time (WT) and Time Spent In System (TSIS)
WT = max(0, WB_TSB(chosen_wash_bay) - AT);
TSIS = WT + WB1_ST + WB2_ST + WB3_ST;
wash_bay(car_no, 5) = AT;  % Update arrival time in the wash_bay matrix
```

WB_TSB(chosen_wash_bay) is the start time of service for the chosen wash bay.
AT is the arrival time of the current car.
The waiting time is calculated as the difference between the start time of service and the arrival time (WB_TSB(chosen_wash_bay) - AT).
max(0, ...) ensures that the waiting time is non-negative. If the car arrives after the service start time, the waiting time is zero.
Time spent in the system is the sum of the waiting time that was calculated previously with WB1_ST, WB2_ST, WB3_ST.
This updates the fifth column of the wash_bay matrix for the current car (car_no) with the arrival time (AT).
This ensures that the wash_bay matrix records the arrival time of each car for later use.

## Random_Service_Time.m

```
2        % Fixed parameters for LCG
3        a = 1664525;
4        c = 1013904223;
5        m = 2^32;
```

Line 3-5 are constants of the Linear Congruential Generator (LCG).
Initial a = 1664525.Initial c = 1013904223.Initial m = 2^32.

```
20      for wash_bay = 1:3
21          % Generate a random seed for each wash bay
22          seed = rand;
23
24          % Generate a random floating-point number between 0 and 1
25          if strcmp(rng_type, 'lcg')
26              random_float = lcg(seed, a, c, m, 1);
27          else
28              random_float = rand();
29          end
30
31          % Scale this number to the desired range and convert it to an integer
32          initial_value = floor(random_float * (max_value - sequence_length + 1 - min_value + 1)) + min_value;
33
34          % Generate the sequence of consecutive integers
35          service_time = initial_value : initial_value + sequence_length - 1;
36
37          % Generate random probabilities and normalize them
38          if strcmp(rng_type, 'lcg')
39              random_probabilities = lcg(seed, a, c, m, sequence_length);
40          else
41              random_probabilities = rand(1, sequence_length);
42      end
```

Line 22  seed = rand; is to generate a random seed for each wash bay and different sequences of random numbers.
Line 25  if strcmp(rng_type, 'lcg') the meaning is compare two strings if the random number generator type is ‘lcg’ then true,otherwise false.
Line 26 random_float = lcg(seed, a, c, m, 1); .When line 25 is true,then call the lcg function to generate a random floating-point number between 0 and 1.
Line 28 random_float = rand();.When line 25 is false,then use the rand() function to generate the random floating-point number between 0 and 1.
Line 32 initial_value = floor(random_float * (max_value - sequence_length + 1 - min_value + 1)) + min_value; the meaning is use the previous random floating-point number between 0 and 1 multiply (max_value - sequence_length + 1 - min_value + 1).Then rounds down to the nearest integer.Then + min_value.And assign the value as initial value.
Line 35 service_time = initial_value : initial_value + sequence_length - 1;the meaning initial_value: is to take the previously calculated starting integer value,initial_value + sequence_length - 1 is the ending value of the sequence,and assign it to service_time.
Line 38 if strcmp(rng_type, 'lcg') the meaning is compare two strings,if the random number generator type is ‘lcg’ then true,otherwise false.
Line 39  random_probabilities = lcg(seed, a, c, m, sequence_length); is when Line 38 is true then call lcg function and use lcg method to generate random number for random_probabilities.The sequence_length is determine how many random numbers need to generate.

Line 41 random_probabilities = rand(1, sequence_length);When line 25 is false,then use rand() function to generate the random number for random_probabilities.

```
65          for i = 1:sequence_length
66              if i > 1
67                  range_start = floor(cdf_wb(i-1) * 1000) + 1;
68              else
69                  range_start = 1;
70              end
71              range_end = floor(cdf_wb(i) * 1000);
72              rn_range_wb{i} = sprintf('%d to %d', range_start, range_end);
73          end
```

Line 65 for i = 1:sequence_length starts the loop that runs from i = 1 to i = sequence_length.
Line 67 range_start = floor(cdf_wb(i-1) * 1000) + 1; is when  Line 66 is true then Line 67 cdf_wb(i-1) is accessing the CDF value at the position i-1 in the cdf_wb array. This is the previous CDF value before the current i.And then take that value multiply 1000.After that,apply floor function to round down the value to the nearest integer.And take that value + 1 assign to range_start.
Line 69  range_start = 1; is when Line 66 is false then range_start is started from 1.
Line 71 range_end = floor(cdf_wb(i) * 1000); is to access the CDF value at the position i in the cdf_wb array and multiply 1000,After that, take that value and apply floor function to round down to the nearest integer ,assign to range_end.
Line 72 rn_range_wb{i} = sprintf('%d to %d', range_start, range_end); the sprintf is a function that take a format string and a list of values to insert into the string.'%d to %d' is the format string.%d is the place to put integer value. range_start, range_end is the integer that i want to put into the format string.and display the range of random numbers to each wash bay .

# Car_Wash_Service_Type.m

```
31      % Calculate the random number ranges based on the CDF
32      st_rn_range = cell(num_types, 1); % Use cell array instead of strings
33      for i = 1:num_types
34          if i > 1
35              range_start = floor(st_cdf(i-1) * 100) + 1;
36          else
37              range_start = 1;
38          end
39          range_end = floor(st_cdf(i) * 100); %example:range_start = 1,range_end = floor(0.18 * 100) = 18
40          st_rn_range{i} = sprintf('%d - %d', range_start, range_end); %example:st_rn_range{1} = 1 - 18
41      end
```

Line 33 for i = 1:num_types is a loop that run from 1 to num_types, which is the number of different car wash service types.
Line 34 if i > 1 is if i is greater than 1.
Line 35 range_start = floor(st_cdf(i-1) * 100) + 1; is when line 34 is true ,then st_cdf(i-1) give the cumulative probability up to the previous service type and multiply 100 and floor the value to

an integer value and + 1 to ensure that the range_start is incremented to avoid overlapping ranges.
Line 37 range_start = 1; is when line 34 is false,then range_start starts from 1.
Line 39 range_end = floor(st_cdf(i) * 100); st_cdf(i) give the cumulative probability up to the current service type and multiply 100 and floor the value to an integer value.
Line 40 st_rn_range{i} = sprintf('%d - %d', range_start, range_end); is display the range start and range end to that cell array st_rn_range.

# Inter_Arrival_Time.m

```
% Ensure that no probability is less than 0.003 and the first probability is not too low or too high
min_prob = 0.01;  % Minimum probability
max_prob = 0.999; % Maximum probability
random_probabilities(random_probabilities < min_prob) = min_prob; % Set lower bound
if random_probabilities(1) < min_prob || random_probabilities(1) > max_prob
    random_probabilities(1) = rand() * (max_prob - min_prob) + min_prob; % Adjust first probability
end
```

It sets a minimum probability (min_prob) of 0.01 and a maximum probability (max_prob) of 0.999. Any probability below the minimum is adjusted to the minimum value. After that, use the if statement to check the first probability. If it falls outside the specified range, random probabilities below the min_prob or  random probabilities more than the max_prob, it is adjusted to a random value within the range from min_prob to max_prob. This ensures that no probability is too small and that the first probability is within acceptable limits.

```
38      % Normalize the probabilities so they sum to 1
39      probabilities = random_probabilities / sum(random_probabilities);
40
41      % Manually round probabilities to three decimal places
42      probabilities = round(probabilities * 1000) / 1000;
43
44      % Ensure the sum of the rounded probabilities is 1 by adjusting the last element
45      probabilities(end) = 1 - sum(probabilities(1:end-1));
46
47      % Calculate the cumulative distribution function (CDF)
48      cdf = cumsum(probabilities);
49
50      % Ensure that CDF ends exactly at 1
51      cdf(end) = 1;
52
53      % Calculate the random number ranges based on the CDF
54      random_number_ranges = cell(sequence_length, 1); % Use cell array instead of strings
```

Line 39, it first normalizes the probabilities so their sum equals 1.
Line 42, rounds each probability to three decimal places
Line 45, after rounding, it adjusts the last probability to ensure the total sum remains exactly 1.
Line 48, it calculates the CDF from the probabilities, which is the cumulative sum of the probabilities.

Line 51, it ensures the last value of the CDF is exactly 1.
Line 54, it calculates the random number ranges based on the CDF values and stores these ranges in a cell array. This helps in mapping random numbers to the corresponding probabilities.

```
    for i = 1:sequence_length
        if i > 1
            range_start = floor(cdf(i-1) * 1000) + 1; % Start of range
        else
            range_start = 1; % First range starts at 1
        end
        range_end = floor(cdf(i) * 1000); % End of range
        random_number_ranges{i} = sprintf('%d to %d', range_start, range_end); % Store range as string
    end
```

Use a for-loop that iterates from 1 to sequence_length, allowing to process each probability interval. After that, use the if else statement to check if 'i' is bigger than 1 or not. If true, range_start = floor(cdf(i-1) * 1000) + 1. If false, the start of the range is set to 1. Then, calculates the end of the current range by taking the current CDF value (cdf(i)), scaling it to the 1-1000 range, and using the floor function to ensure it is an integer. Then, generate and store the ranges of random numbers for each probability interval, which are used to map random numbers to specific inter-arrival times based on the computed probabilities.

## calculate_averages.m

```
% Calculate average waiting time
if total_waiting_time > 0
    average_waiting_time = total_waiting_time / num_cars_with_waiting_time; % Compute average waiting time if there is any waiting time
else
    average_waiting_time = 0; % Set average waiting time to 0 if there is no waiting time
end
```

If there is a positive total waiting time (total_waiting_time > 0), it computes the average waiting time by dividing the total waiting time by the number of cars that had to wait (num_cars_with_waiting_time). If there is no waiting time, it sets the average waiting time to 0.

```
% Calculate average inter-arrival time
if num_cars > 1
    average_inter_arrival_time = total_inter_arrival_time / (num_cars - 1); % Compute average inter-arrival time if there is more than one car
else
    average_inter_arrival_time = 0; % Set average inter-arrival time to 0 if there is only one or no car
end
```

If there is more than one car (num_cars > 1), it computes the average inter-arrival time by dividing the total inter-arrival time (total_inter_arrival_time) by the number of gaps between cars, which is num_cars - 1. If there is only one car or no cars, it sets the average inter-arrival time to 0.

```
% Calculate average arrival time
if num_cars > 1
    average_arrival_time = total_arrival_time / num_cars; % Compute average arrival time if there is more than one car
else
    average_arrival_time = 0; % Set average arrival time to 0 if there is only one or no car
end
```

If there is at least one car (num_cars > 0), it computes the average arrival time by dividing the total arrival time (total_arrival_time) by the number of cars (num_cars). If there are no cars, it sets the average arrival time to 0.

```

% Calculate average time spent in the system per car
if num_cars > 0
    average_time_spent_in_system = total_time_spent_in_system / num_cars; % Compute average time spent in the system per car if there is at least one car
else
    average_time_spent_in_system = 0; % Set average time spent in the system to 0 if there are no cars
end
```

If there is at least one car (num_cars > 0), it computes the average time spent in the system by dividing the total time spent in the system (total_time_spent_in_system) by the number of cars (num_cars). If there are no cars, it sets the average time spent in the system to 0.

```
% Calculate probability that a car owner has to wait in the queue
if num_cars > 0
    probability_waiting = num_cars_with_waiting_time / num_cars; % Compute probability of waiting if there is at least one car
else
    probability_waiting = 0;  % Default to 0 if there are no cars
end
```

If there is at least one car (num_cars > 0), it computes the probability of waiting by dividing the number of cars that had to wait (num_cars_with_waiting_time) by the total number of cars (num_cars). If there are no cars, it sets the probability of waiting to 0.

```
% Calculate average service time per wash bay
if num_cars_served_WB1 > 0
    average_service_time_WB1 = total_service_time_WB1 / num_cars_served_WB1; % Compute average service time for Wash Bay 1 if there were cars served
else
    average_service_time_WB1 = 0;  % Set to zero if no cars were served by Wash Bay 1
end

if num_cars_served_WB2 > 0
    average_service_time_WB2 = total_service_time_WB2 / num_cars_served_WB2; % Compute average service time for Wash Bay 2 if there were cars served
else
    average_service_time_WB2 = 0;  % Set to zero if no cars were served by Wash Bay 2
end

if num_cars_served_WB3 > 0
    average_service_time_WB3 = total_service_time_WB3 / num_cars_served_WB3; % Compute average service time for Wash Bay 3 if there were cars served
else
    average_service_time_WB3 = 0;  % Set to zero if no cars were served by Wash Bay 3
end
```

if there were any cars served by Wash Bay 1 (num_cars_served_WB1 > 0). If there were cars served, it computes the average service time by dividing the total service time spent on Wash Bay 1 (total_service_time_WB1) by the number of cars served (num_cars_served_WB1). If no cars were served by Wash Bay 1, it sets the average service time to 0.
This if else statement is the same as WB2 and WB3.

```
    % Calculate percentage of time each wash bay was busy
    if total_simulated_time_WB1 > 0
        percentage_time_WB1_busy = (total_service_time_WB1 / total_simulated_time_WB1) * 100;
    else
        percentage_time_WB1_busy = 0;
    end

    if total_simulated_time_WB2 > 0
        percentage_time_WB2_busy = (total_service_time_WB2 / total_simulated_time_WB2) * 100;
    else
        percentage_time_WB2_busy = 0;
    end

    if total_simulated_time_WB3 > 0
        percentage_time_WB3_busy = (total_service_time_WB3 / total_simulated_time_WB3) * 100;
    else
        percentage_time_WB3_busy = 0;
    end
```

If there was any simulated time (total_simulated_time_WB1 > 0). If so, it computes the percentage of time WB1 was actively serving (total_service_time_WB1) out of the total simulated time and converts it to a percentage. If no simulated time is recorded (total_simulated_time_WB1 <= 0), it sets the busy percentage to 0 to avoid division errors.This if else statement is the same as WB2 and WB3.

# Generate Car Number.m

```
function wash_bay = Generate_Car_Numbers(num_cars, rng_type)
    % Generate a sequence of integers from 1 to num_cars
    car_numbers = 1:num_cars;  % Create an array of car numbers

    % Generate wash bay table with additional columns for RN Service Time and RN Service Type
    wash_bay = zeros(num_cars, 5);  % Initialize a matrix to store car numbers and random values
    wash_bay(:, 1) = car_numbers;  % First column: Car numbers
```

This section creates an array of car numbers ranging from 1 to num_cars and initializes a matrix wash_bay with five columns to store this data and additional attributes for each car. The first column is filled with car numbers.

```
% First row inter-arrival time is 0
wash_bay(1, 2) = 0;  % Set the inter-arrival time of the first car to 0
```

The inter-arrival time for the first car is set to zero, as there's no car before it to measure the time from.

```
% Generate random inter-arrival times (between 1 and 1000) for remaining rows
for i = 2:num_cars
    wash_bay(i, 2) = floor(rand * 1000) + 1;  % Generate and assign random inter-arrival times
end
```

For each car from the second to the last, this loop generates a random inter-arrival time between 1 and 1000 milliseconds and stores it in the second column of the wash_bay matrix.

```
% Generate random RN Service Time (between 1 and 1000) for all rows
for i = 1:num_cars
    wash_bay(i, 3) = floor(rand * 1000) + 1;  % Generate and assign random service times using default RNG
end
```

The service times are generated within the range of 1 to 1000.
The generated service times are assigned to the third column of the wash_bay matrix for each car.
The +1 in floor(rand * 1000) + 1 ensures that the generated random number falls within the inclusive range of 1 to 1000. Without the +1, the range would be 0 to 999. By adding 1, the formula adjusts the range to 1 to 1000, matching the expected range for service times or other random values.

```
% Generate random RN Service Type (between 1 and 100) for all rows
for i = 1:num_cars
    wash_bay(i, 4) = floor(rand * 100) + 1;  % Generate and assign random service types
end
```

This loop assigns a random service type (from 1 to 100) to each car and stores it in the fourth column:
The loop starts with i = 1 and iterates up to the number of cars (num_cars). Each iteration corresponds to one car. Within the loop, a random number is generated for each car using the

rand function. The rand function generates a floating-point number between 0 and 1. This number is then scaled by multiplying it by 100 to shift the range to 0 to 100.

**floor(rand * 100) + 1**

- rand * 100 generates a number between 0 and 100.
- floor(...) rounds this number down to the nearest integer, which results in an integer between 0 and 99.
- Adding 1 shifts this range to become 1 to 100.

The calculated random service type is assigned to the fourth column of the wash_bay matrix for the current car (i). The wash_bay matrix is being used to store all the necessary information about each car, including their assigned random service types.

Once all cars have been assigned a random service type, the loop concludes.

```
wash_bay(:, 5) = 0;
```

This line initializes the fifth column of the wash_bay matrix with zeros.

```
% Determine the maximum width needed for car numbers, inter-arrival times, RN Service Time, and RN Service Type
max_car_width = max(ceil(log10(wash_bay(:, 1) + 1)));  % Calculate the width for car numbers
max_time_width = max(ceil(log10(wash_bay(:, 2) + 1)));  % Calculate the width for inter-arrival times
max_st_width = max(ceil(log10(wash_bay(:, 3) + 1)));  % Calculate the width for service times
max_stype_width = max(ceil(log10(wash_bay(:, 4) + 1)));  % Calculate the width for service types
```

**max_car_width**
Taking the logarithm base 10 of each car number (wash_bay(:, 1)) plus one (to handle cases where a car number could be a power of 10, which would increase its digit count), then applying ‘ceil’ to round up to the nearest integer and lastly, using the max to find the largest value among these, which represents the maximum width required.

**Max_time_width**
It performs the logarithmic calculation on the inter-arrival times (wash_bay(:, 2)), rounds up, and then determines the maximum width required.

For example,
wash_bay =

| | | | | |
|---|---|---|---|---|
| 1 | 0 | 587 | 63 | 0 |
| 2 | 892 | 451 | 97 | 0 |
| 3 | 145 | 912 | 21 | 0 |
| 4 | 763 | 333 | 84 | 0 |
| 5 | 376 | 18 | 54 | 0 |

The inter-arrival times are in the second column of the wash_bay matrix:
0, 892, 145, 763, 376

Add 1 to Each Inter-Arrival Time
0 + 1 = 1
892 + 1 = 893
145 + 1 = 146
763 + 1 = 764
376 + 1 = 377

Apply log10 to Each Number
log10(1) ≈ 0
log10(893) ≈ 2.9514
log10(146) ≈ 2.1644
log10(764) ≈ 2.8837
log10(377) ≈ 2.5763

Apply ceil to Each Result
ceil(0) = 0
ceil(2.9514) = 3
ceil(2.1644) = 3
ceil(2.8837) = 3
ceil(2.5763) = 3

Find the Maximum Value
max(0, 3, 3, 3, 3) = 3

**Max_st_width**
Takes the logarithm base 10 of service times (wash_bay(:, 3)), adds one, rounds up the results, and finds the maximum value.

**Max_stype_width**
Applies the same logarithmic and rounding method to the service types (`wash_bay(:, 4)`), then finds the maximum width.

# LCG.M

```
function random numbers = lcg(seed, a, c, m, n)
```

Defines a function called lcg that returns an array random_numbers. It accepts five parameters:

- seed: Initial value to start the random number generation.
- a: The multiplier.
- c: The increment.
- m: The modulus.
- n: The number of random numbers to generate.

```
random_numbers = rand(1, n); % Preallocate array for random numbers
X = seed; % Initialize the first value with the seed
```

Initializes the random_numbers array with random values to ensure it has the appropriate size for the number of values n that will be generated.
Sets the initial value of X to seed, which is used to start the random number generation process.

```
for i = 1:n
    X = mod(a * X + c, m);  % Generate the next random number using the LCG formula
    random_numbers(i) = X;
end
```

This loop iterates n times to generate n random numbers.
X is updated in each iteration using the formula (a * X + c) % m, which is the core of the Linear Congruential Generator algorithm.
The result of each calculation is then stored in the corresponding index of the random_numbers array.

```
% Normalize to [0, 1]
random_numbers = random_numbers / m;
```

After all random numbers are generated, they are normalized by dividing each by m to scale them into the range [0, 1], making them useful for the simulation.

rand function from FreeMat :

```
--> main
Choose the type of random number generator (e.g., 'rand', 'lcg'): rand

 WB.ST = Wash Bay Service Time
 WB.ST.Prob = Wash Bay Service Time Probability
 WB.ST.CDF = Wash Bay Service Time Cumulative Distribution Function
 WB.ST.Range = Wash Bay Service Time Range

Wash Bay 1
---------------------------------------------------
 WB1.ST | WB1.ST.Prob | WB1.ST.CDF | WB1.ST.Range
---------------------------------------------------
 20      | 0.083        | 0.083       | 1 to 83
 21      | 0.168        | 0.251       | 84 to 251
 22      | 0.061        | 0.312       | 252 to 312
 23      | 0.232        | 0.544       | 313 to 544
 24      | 0.249        | 0.793       | 545 to 793
 25      | 0.207        | 1.000       | 794 to 1000
---------------------------------------------------

Wash Bay 2
---------------------------------------------------
 WB2.ST | WB2.ST.Prob | WB2.ST.CDF | WB2.ST.Range
---------------------------------------------------
 25      | 0.264        | 0.264       | 1 to 264
 26      | 0.208        | 0.472       | 265 to 472
 27      | 0.139        | 0.611       | 473 to 611
 28      | 0.130        | 0.741       | 612 to 741
 29      | 0.245        | 0.986       | 742 to 986
 30      | 0.014        | 1.000       | 987 to 1000
---------------------------------------------------

Wash Bay 3
---------------------------------------------------
 WB3.ST | WB3.ST.Prob | WB3.ST.CDF | WB3.ST.Range
---------------------------------------------------
 17      | 0.227        | 0.227       | 1 to 227
 18      | 0.081        | 0.308       | 228 to 308
 19      | 0.171        | 0.479       | 309 to 479
 20      | 0.267        | 0.746       | 480 to 746
 21      | 0.062        | 0.808       | 747 to 808
 22      | 0.192        | 1.000       | 809 to 1000
---------------------------------------------------
```

```
 IAT = Inter-Arrival Time
 IAT.Prob = Inter-Arrival Time Probability
 IAT.CDF = Inter-Arrival Time Cumulative Distribution Function
 IAT.Range = Inter-Arrival Time Range

---------------------------------------
IAT | IAT.Prob | IAT.CDF  | IAT.Range
---------------------------------------
2   | 0.056    | 0.056    | 1 to 56
3   | 0.019    | 0.075    | 57 to 75
4   | 0.282    | 0.357    | 76 to 357
5   | 0.311    | 0.668    | 358 to 667
6   | 0.239    | 0.907    | 668 to 906
7   | 0.093    | 1.000    | 907 to 1000
---------------------------------------


 Service Type = Car Wash Service Type
 Prob = Car Wash Service Type Probabitlty
 CDF = Car Wash Service Type Cumulative Distribution Function
 Range = Car Wash Service Type Range

-----------------------------------------
Service Type  | Prob | CDF  | Range
-----------------------------------------
 Basic Wash   | 0.36 | 0.36 | 1 - 36
 Deluxe Wash  | 0.19 | 0.55 | 37 - 55
 Premium Wash | 0.45 | 1.00 | 56 - 100
-----------------------------------------
```

Enter the number of cars: 10

C.No = Car Number
RN.IAT = Random Number Inter-Arrival Time
IAT = Inter-Arrival Time
AT = Arrival Time
RN.STy = Random Number Service Type
SType = Service Type
RN.ST = Random Number Service Time
WB1.ST = Wash Bay 1 Service Time
WB1.TSB = Wash Bay 1 Time Service Begins
WB1.TSE = Wash Bay 1 Time Service Ends
WB2.ST = Wash Bay 2 Service Time
WB2.TSB = Wash Bay 2 Time Service Begins
WB2.TSE = Wash Bay 2 Time Service Ends
WB3.ST = Wash Bay 3 Service Time
WB3.TSB = Wash Bay 3 Time Service Begins
WB3.TSE = Wash Bay 3 Time Service Ends
WT = Waiting Time
TSIS = Time Spent In The System

| C.No | RN.IAT | IAT | AT | RN.STy | SType | RN.ST | WB1.ST | WB1.TSB | WB1.TSE | WB2.ST | WB2.TSB | WB2.TSE | WB3.ST | WB3.TSB | WB3.TSE | WT | TSIS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 79 | Premium Wash | 435 | 23 | 0 | 23 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 23 |
| 2 | 107 | 4 | 4 | 90 | Premium Wash | 737 | 0 | 0 | 0 | 28 | 4 | 32 | 0 | 0 | 0 | 0 | 28 |
| 3 | 827 | 6 | 10 | 8 | Basic Wash | 372 | 0 | 0 | 0 | 0 | 0 | 0 | 19 | 10 | 29 | 0 | 19 |
| 4 | 294 | 4 | 14 | 6 | Basic Wash | 943 | 25 | 23 | 48 | 0 | 0 | 0 | 0 | 0 | 0 | 9 | 34 |
| 5 | 710 | 6 | 20 | 4 | Basic Wash | 419 | 0 | 0 | 0 | 0 | 0 | 0 | 19 | 29 | 48 | 9 | 28 |
| 6 | 559 | 5 | 25 | 3 | Basic Wash | 320 | 0 | 0 | 0 | 26 | 32 | 58 | 0 | 0 | 0 | 7 | 33 |
| 7 | 434 | 5 | 30 | 55 | Deluxe Wash | 270 | 22 | 48 | 70 | 0 | 0 | 0 | 0 | 0 | 0 | 18 | 40 |
| 8 | 904 | 6 | 36 | 95 | Premium Wash | 693 | 0 | 0 | 0 | 0 | 0 | 0 | 20 | 48 | 68 | 12 | 32 |
| 9 | 976 | 7 | 43 | 12 | Basic Wash | 86 | 0 | 0 | 0 | 25 | 58 | 83 | 0 | 0 | 0 | 15 | 40 |
| 10 | 440 | 5 | 48 | 33 | Basic Wash | 533 | 0 | 0 | 0 | 0 | 0 | 0 | 20 | 68 | 88 | 20 | 40 |

Average Waiting Time of Car Owners: 12.86
Average Inter-Arrival Time: 5.33
Average Arrival Time: 23.00
Average Time Spent in the System per Car: 31.70
Probability that a car owner has to wait in the queue: 0.70
Average Service Time for Wash Bay 1: 23.33
Average Service Time for Wash Bay 2: 26.33
Average Service Time for Wash Bay 3: 19.50
Percentage of Time WB1 was Busy: 100.00%
Percentage of Time WB2 was Busy: 95.18%
Percentage of Time WB3 was Busy: 88.64%

linear congruential generators :

```
--> main
Choose the type of random number generator (e.g., 'rand', 'lcg'): lcg

 WB.ST = Wash Bay Service Time
 WB.ST.Prob = Wash Bay Service Time Probability
 WB.ST.CDF = Wash Bay Service Time Cumulative Distribution Function
 WB.ST.Range = Wash Bay Service Time Range

Wash Bay 1
---------------------------------------------------
 WB1.ST | WB1.ST.Prob | WB1.ST.CDF | WB1.ST.Range
---------------------------------------------------
 17     | 0.117       | 0.117      | 1 to 117
 18     | 0.236       | 0.353      | 118 to 353
 19     | 0.045       | 0.398      | 354 to 397
 20     | 0.333       | 0.731      | 398 to 731
 21     | 0.241       | 0.972      | 732 to 972
 22     | 0.028       | 1.000      | 973 to 1000
---------------------------------------------------

Wash Bay 2
---------------------------------------------------
 WB2.ST | WB2.ST.Prob | WB2.ST.CDF | WB2.ST.Range
---------------------------------------------------
 17     | 0.058       | 0.058      | 1 to 58
 18     | 0.179       | 0.237      | 59 to 237
 19     | 0.186       | 0.423      | 238 to 423
 20     | 0.188       | 0.611      | 424 to 611
 21     | 0.158       | 0.769      | 612 to 769
 22     | 0.231       | 1.000      | 770 to 1000
---------------------------------------------------

Wash Bay 3
---------------------------------------------------
 WB3.ST | WB3.ST.Prob | WB3.ST.CDF | WB3.ST.Range
---------------------------------------------------
 17     | 0.068       | 0.068      | 1 to 68
 18     | 0.161       | 0.229      | 69 to 229
 19     | 0.247       | 0.476      | 230 to 476
 20     | 0.110       | 0.586      | 477 to 586
 21     | 0.192       | 0.778      | 587 to 778
 22     | 0.222       | 1.000      | 779 to 1000
---------------------------------------------------
```

```
 IAT = Inter-Arrival Time
 IAT.Prob = Inter-Arrival Time Probability
 IAT.CDF = Inter-Arrival Time Cumulative Distribution Function
 IAT.Range = Inter-Arrival Time Range

---------------------------------------
IAT | IAT.Prob | IAT.CDF  | IAT.Range
---------------------------------------
1   | 0.108    | 0.108    | 1 to 108
2   | 0.043    | 0.151    | 109 to 151
3   | 0.225    | 0.376    | 152 to 376
4   | 0.243    | 0.619    | 377 to 619
5   | 0.371    | 0.990    | 620 to 990
6   | 0.010    | 1.000    | 991 to 1000
---------------------------------------


 Service Type = Car Wash Service Type
 Prob = Car Wash Service Type Probabitlty
 CDF = Car Wash Service Type Cumulative Distribution Function
 Range = Car Wash Service Type Range

-----------------------------------------
Service Type   | Prob | CDF  | Range
-----------------------------------------
 Basic Wash    | 0.07 | 0.07 | 1 - 7
 Deluxe Wash   | 0.88 | 0.95 | 8 - 95
 Premium Wash  | 0.05 | 1.00 | 96 - 100
-----------------------------------------
```

Enter the number of cars: 10

C.No = Car Number
RN.IAT = Random Number Inter-Arrival Time
IAT = Inter-Arrival Time
AT = Arrival Time
RN.STy = Random Number Service Type
SType = Service Type
RN.ST = Random Number Service Time
WB1.ST = Wash Bay 1 Service Time
WB1.TSB = Wash Bay 1 Time Service Begins
WB1.TSE = Wash Bay 1 Time Service Ends
WB2.ST = Wash Bay 2 Service Time
WB2.TSB = Wash Bay 2 Time Service Begins
WB2.TSE = Wash Bay 2 Time Service Ends
WB3.ST = Wash Bay 3 Service Time
WB3.TSB = Wash Bay 3 Time Service Begins
WB3.TSE = Wash Bay 3 Time Service Ends
WT = Waiting Time
TSIS = Time Spent In The System

| C.No | RN.IAT | IAT | AT | RN.STy | SType | RN.ST | WB1.ST | WB1.TSB | WB1.TSE | WB2.ST | WB2.TSB | WB2.TSE | WB3.ST | WB3.TSB | WB3.TSE | WT | TSIS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 82 | Deluxe Wash | 363 | 19 | 0 | 19 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 19 |
| 2 | 873 | 5 | 5 | 76 | Deluxe Wash | 22 | 0 | 0 | 0 | 17 | 5 | 22 | 0 | 0 | 0 | 0 | 17 |
| 3 | 718 | 5 | 10 | 18 | Deluxe Wash | 503 | 0 | 0 | 0 | 0 | 0 | 0 | 20 | 10 | 30 | 0 | 20 |
| 4 | 948 | 5 | 15 | 50 | Deluxe Wash | 888 | 21 | 19 | 40 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 25 |
| 5 | 71 | 1 | 16 | 33 | Deluxe Wash | 272 | 0 | 0 | 0 | 19 | 22 | 41 | 0 | 0 | 0 | 6 | 25 |
| 6 | 765 | 5 | 21 | 10 | Deluxe Wash | 767 | 0 | 0 | 0 | 0 | 0 | 0 | 21 | 30 | 51 | 9 | 30 |
| 7 | 409 | 4 | 25 | 57 | Deluxe Wash | 620 | 20 | 40 | 60 | 0 | 0 | 0 | 0 | 0 | 0 | 15 | 35 |
| 8 | 944 | 5 | 30 | 13 | Deluxe Wash | 829 | 0 | 0 | 0 | 22 | 41 | 63 | 0 | 0 | 0 | 11 | 33 |
| 9 | 860 | 5 | 35 | 88 | Deluxe Wash | 877 | 0 | 0 | 0 | 0 | 0 | 0 | 22 | 51 | 73 | 16 | 38 |
| 10 | 937 | 5 | 40 | 57 | Deluxe Wash | 106 | 17 | 60 | 77 | 0 | 0 | 0 | 0 | 0 | 0 | 20 | 37 |

Average Waiting Time of Car Owners: 11.57
Average Inter-Arrival Time: 4.44
Average Arrival Time: 19.70
Average Time Spent in the System per Car: 27.90
Probability that a car owner has to wait in the queue: 0.70
Average Service Time for Wash Bay 1: 19.25
Average Service Time for Wash Bay 2: 19.33
Average Service Time for Wash Bay 3: 21.00
Percentage of Time WB1 was Busy: 100.00%
Percentage of Time WB2 was Busy: 92.06%
Percentage of Time WB3 was Busy: 86.30%