LAPORAN TUGAS BESAR I

IF4021 PEMODELAN DAN SIMULASI

Oleh:

Doddy Aditya Wiranugraha 13517008

PROGRAM STUDI TEKNIK INFORMATIKA

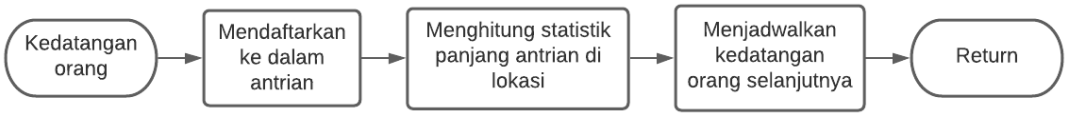
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

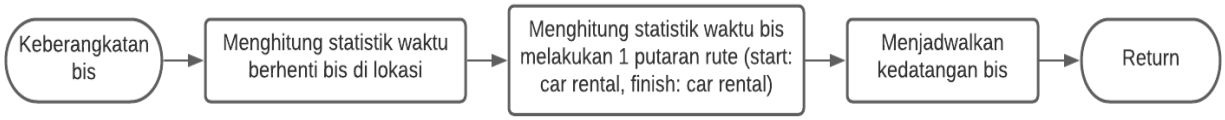
2021

I. Deskripsi Persoalan dan Solusi

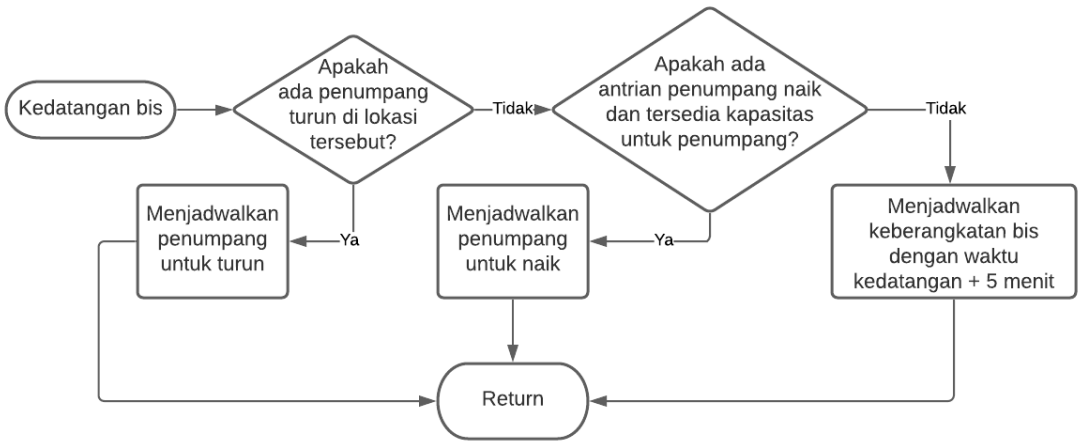
1. Event Kedatangan Orang



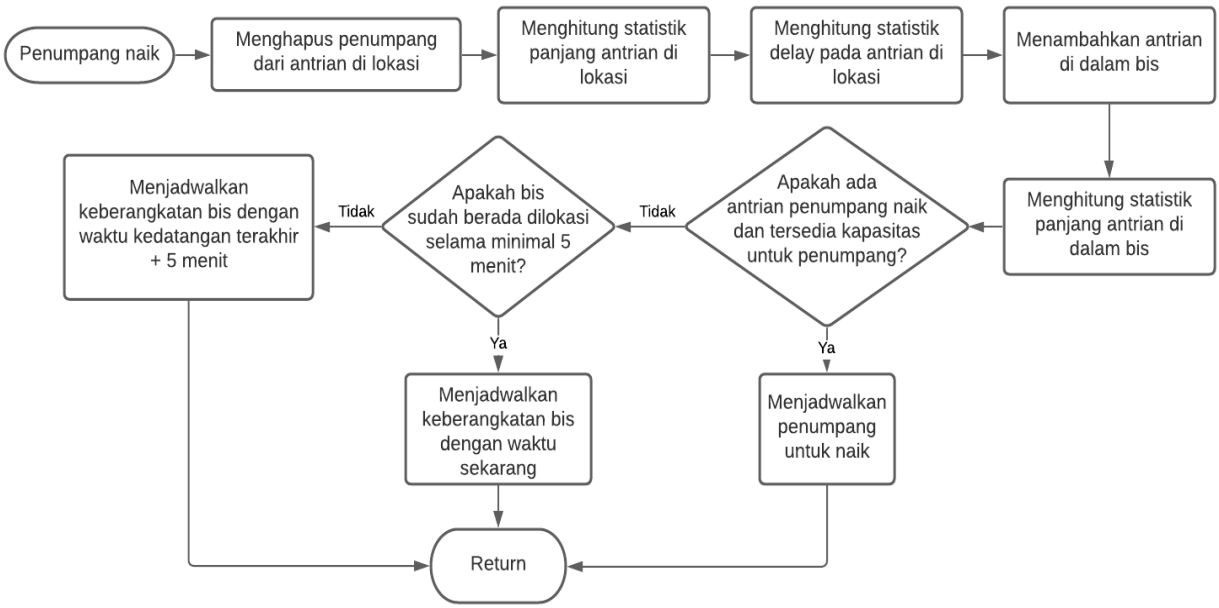
2. Event Keberangkatan Bis



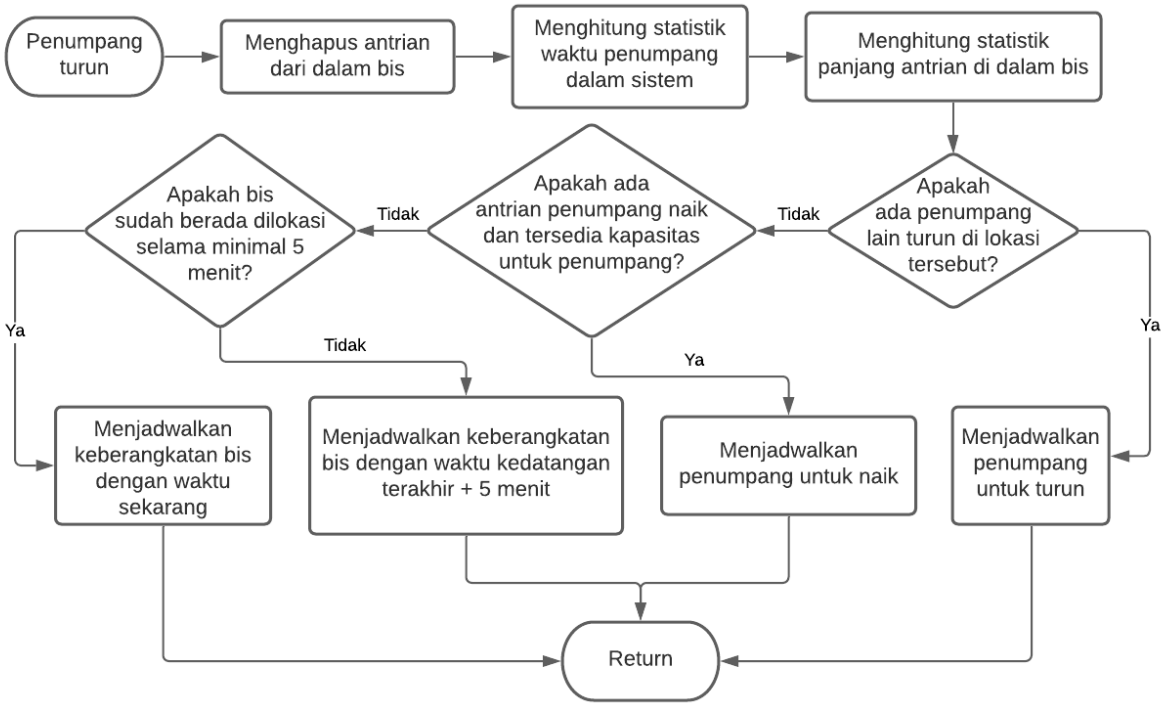
3. Event Kedatangan Bis



4. Event Penumpang Menaiki Bis



5. Event Penumpang Turun dari Bis



II. Source Code

a. car_rental.in

```
3 2 80
14 10 24
30
0.583 1.00
16 24
15 25
0 1 0
0 0 4.5
4.5 0 0
5
```

b. car_rental.c

```c
#include "simlib.h"

#define EVENT_PERSON_ARRIVAL 1
#define EVENT_BUS_ARRIVAL 2
#define EVENT_BUS_DEPARTURE 3
#define EVENT_UNLOADING 4
#define EVENT_LOADING 5
#define EVENT_END_SIMULATION 6

#define STREAM_INTERARRIVAL_1 1
```

```
#define STREAM_INTERARRIVAL_2 2
#define STREAM_INTERARRIVAL_3 3
#define STREAM_UNLOADING 4
#define STREAM_LOADING 5
#define STREAM_DESTINATION 6

#define LIST_TERMINAL_1 1
#define LIST_TERMINAL_2 2
#define LIST_CAR_RENTAL 3
#define LIST_TO_TERMINAL_1 4
#define LIST_TO_TERMINAL_2 5
#define LIST_TO_CAR_RENTAL 6

#define VARIABLE_TERMINAL_1 1
#define VARIABLE_TERMINAL_2 2
#define VARIABLE_CAR_RENTAL 3
#define VARIABLE_DELAY_TERMINAL_1 4
#define VARIABLE_DELAY_TERMINAL_2 5
#define VARIABLE_DELAY_CAR_RENTAL 6
#define VARIABLE_BUS 7
#define VARIABLE_PERSON 8
#define VARIABLE_BUS_STOP_TERMINAL_1 9
#define VARIABLE_BUS_STOP_TERMINAL_2 10
#define VARIABLE_BUS_STOP_CAR_RENTAL 11
#define VARIABLE_BUS_LOOP 12

#define TERMINAL_1 1
#define TERMINAL_2 2
#define CAR_RENTAL 3
#define MAX_NUM_LOCATION 3
#define MAX_RANGE 2
#define MAX_BUS_CAPACITY 20

int num_location, num_terminal, bus_capacity, bus_start_location, bus_stop_location;
double person_first_arrival, bus_last_arrival, bus_last_departure, bus_speed, bus_last_arrival,
 bus_min_time_process, length_simulation, prob_distrib_terminal[26], interarrival_time[MAX_NUM_
LOCATION + 1],
       load_uniform_distrib_terminal[MAX_RANGE + 1], unload_uniform_distrib_terminal[MAX_RANGE
+ 1],
       route_distance[MAX_NUM_LOCATION + 1][MAX_NUM_LOCATION + 1];
FILE *infile, *outfile;

int recent_bus_capacity()
{
  return (list_size[LIST_TO_TERMINAL_1] + list_size[LIST_TO_TERMINAL_2] + list_size[LIST_TO_CAR
_RENTAL]);
}
```

```
void person_arrive(void)
{
  if (transfer[3] == TERMINAL_1)
  {
    list_file(LAST, LIST_TERMINAL_1);
    timest(list_size[LIST_TERMINAL_1], VARIABLE_TERMINAL_1);
    transfer[3] = TERMINAL_1;
    event_schedule(sim_time + expon(1 / interarrival_time[TERMINAL_1], STREAM_INTERARRIVAL_1),
EVENT_PERSON_ARRIVAL);
  }
  else if (transfer[3] == TERMINAL_2)
  {
    list_file(LAST, LIST_TERMINAL_2);
    timest(list_size[LIST_TERMINAL_2], VARIABLE_TERMINAL_2);
    transfer[3] = TERMINAL_2;
    event_schedule(sim_time + expon(1 / interarrival_time[TERMINAL_2], STREAM_INTERARRIVAL_2),
EVENT_PERSON_ARRIVAL);
  }
  else
  {
    if (random_integer(prob_distrib_terminal, STREAM_DESTINATION) == 1)
    {
      transfer[3] = TERMINAL_1;
    }
    else
    {
      transfer[3] = TERMINAL_2;
    }
    list_file(LAST, LIST_CAR_RENTAL);
    timest(list_size[LIST_CAR_RENTAL], VARIABLE_CAR_RENTAL);
    transfer[3] = CAR_RENTAL;
    event_schedule(sim_time + expon(1 / interarrival_time[CAR_RENTAL], STREAM_INTERARRIVAL_3),
EVENT_PERSON_ARRIVAL);
  }
}

void bus_arrive(void)
{
  bus_last_arrival = transfer[1];
  if (transfer[3] == TERMINAL_1)
  {
    transfer[3] = TERMINAL_1;
    if (list_size[LIST_TO_TERMINAL_1] > 0)
    {
      event_schedule(sim_time + uniform(unload_uniform_distrib_terminal[1], unload_uniform_dist
rib_terminal[2], STREAM_UNLOADING), EVENT_UNLOADING);
    }
    else if (list_size[LIST_TERMINAL_1] > 0 && recent_bus_capacity() < MAX_BUS_CAPACITY)
```

```
    {
      event_schedule(sim_time + uniform(load_uniform_distrib_terminal[1], load_uniform_distrib_
terminal[2], STREAM_LOADING), EVENT_LOADING);
    }
    else
    {
      event_schedule(bus_last_arrival + bus_min_time_process / 60, EVENT_BUS_DEPARTURE);
    }
  }
  else if (transfer[3] == TERMINAL_2)
  {
    transfer[3] = TERMINAL_2;
    if (list_size[LIST_TO_TERMINAL_2] > 0)
    {
      event_schedule(sim_time + uniform(unload_uniform_distrib_terminal[1], unload_uniform_dist
rib_terminal[2], STREAM_UNLOADING), EVENT_UNLOADING);
    }
    else if (list_size[LIST_TERMINAL_2] > 0 && recent_bus_capacity() < MAX_BUS_CAPACITY)
    {
      event_schedule(sim_time + uniform(load_uniform_distrib_terminal[1], load_uniform_distrib_
terminal[2], STREAM_LOADING), EVENT_LOADING);
    }
    else
    {
      event_schedule(bus_last_arrival + bus_min_time_process / 60, EVENT_BUS_DEPARTURE);
    }
  }
  else
  {
    transfer[3] = CAR_RENTAL;
    if (list_size[LIST_TO_CAR_RENTAL] > 0)
    {
      event_schedule(sim_time + uniform(unload_uniform_distrib_terminal[1], unload_uniform_dist
rib_terminal[2], STREAM_UNLOADING), EVENT_UNLOADING);
    }
    else if (list_size[LIST_CAR_RENTAL] > 0 && recent_bus_capacity() < MAX_BUS_CAPACITY)
    {
      event_schedule(sim_time + uniform(load_uniform_distrib_terminal[1], load_uniform_distrib_
terminal[2], STREAM_LOADING), EVENT_LOADING);
    }
    else
    {
      event_schedule(bus_last_arrival + bus_min_time_process / 60, EVENT_BUS_DEPARTURE);
    }
  }
}

void bus_depart(void)
```

```
{
  if (transfer[3] == TERMINAL_1)
  {
    sampst(sim_time - bus_last_arrival, VARIABLE_BUS_STOP_TERMINAL_1);
    transfer[3] = TERMINAL_2;
    event_schedule(sim_time + route_distance[TERMINAL_1][TERMINAL_2] / bus_speed, EVENT_BUS_ARR
IVAL);
  }
  else if (transfer[3] == TERMINAL_2)
  {
    sampst(sim_time - bus_last_arrival, VARIABLE_BUS_STOP_TERMINAL_2);
    transfer[3] = CAR_RENTAL;
    event_schedule(sim_time + route_distance[TERMINAL_2][CAR_RENTAL] / bus_speed, EVENT_BUS_ARR
IVAL);
  }
  else
  {
    sampst(sim_time - bus_last_departure, VARIABLE_BUS_LOOP);
    bus_last_departure = sim_time;
    sampst(sim_time - bus_last_arrival, VARIABLE_BUS_STOP_CAR_RENTAL);
    transfer[3] = TERMINAL_1;
    event_schedule(sim_time + route_distance[CAR_RENTAL][TERMINAL_1] / bus_speed, EVENT_BUS_ARR
IVAL);
  }
}

void unload(void)
{
  if (transfer[3] == TERMINAL_1)
  {
    list_remove(FIRST, LIST_TO_TERMINAL_1);
    sampst(sim_time - transfer[1], VARIABLE_PERSON);
    timest(recent_bus_capacity(), VARIABLE_BUS);
    transfer[3] = TERMINAL_1;
    if (list_size[LIST_TO_TERMINAL_1] > 0)
    {
      event_schedule(sim_time + uniform(unload_uniform_distrib_terminal[1], unload_uniform_dist
rib_terminal[2], STREAM_UNLOADING), EVENT_UNLOADING);
    }
    else if (list_size[LIST_TERMINAL_1] > 0 && recent_bus_capacity() < MAX_BUS_CAPACITY)
    {
      event_schedule(sim_time + uniform(load_uniform_distrib_terminal[1], load_uniform_distrib_
terminal[2], STREAM_LOADING), EVENT_LOADING);
    }
    else
    {
      if ((sim_time - bus_last_arrival) < (bus_min_time_process / 60))
      {
```

```
        event_schedule(bus_last_arrival + bus_min_time_process / 60, EVENT_BUS_DEPARTURE);
      }
      else
      {
        event_schedule(sim_time, EVENT_BUS_DEPARTURE);
      }
    }
  }
  else if (transfer[3] == TERMINAL_2)
  {
    list_remove(FIRST, LIST_TO_TERMINAL_2);
    sampst(sim_time - transfer[1], VARIABLE_PERSON);
    timest(recent_bus_capacity(), VARIABLE_BUS);
    transfer[3] = TERMINAL_2;
    if (list_size[LIST_TO_TERMINAL_2] > 0)
    {
      event_schedule(sim_time + uniform(unload_uniform_distrib_terminal[1], unload_uniform_dist
rib_terminal[2], STREAM_UNLOADING), EVENT_UNLOADING);
    }
    else if (list_size[LIST_TERMINAL_2] > 0 && recent_bus_capacity() < MAX_BUS_CAPACITY)
    {
      event_schedule(sim_time + uniform(load_uniform_distrib_terminal[1], load_uniform_distrib_
terminal[2], STREAM_LOADING), EVENT_LOADING);
    }
    else
    {
      if ((sim_time - bus_last_arrival) < (bus_min_time_process / 60))
      {
        event_schedule(bus_last_arrival + bus_min_time_process / 60, EVENT_BUS_DEPARTURE);
      }
      else
      {
        event_schedule(sim_time, EVENT_BUS_DEPARTURE);
      }
    }
  }
  else
  {
    list_remove(FIRST, LIST_TO_CAR_RENTAL);
    sampst(sim_time - transfer[1], VARIABLE_PERSON);
    timest(recent_bus_capacity(), VARIABLE_BUS);
    transfer[3] = CAR_RENTAL;
    if (list_size[LIST_TO_CAR_RENTAL] > 0)
    {
      event_schedule(sim_time + uniform(unload_uniform_distrib_terminal[1], unload_uniform_dist
rib_terminal[2], STREAM_UNLOADING), EVENT_UNLOADING);
    }
    else if (list_size[LIST_CAR_RENTAL] > 0 && recent_bus_capacity() < MAX_BUS_CAPACITY)
```

```
    {
      event_schedule(sim_time + uniform(load_uniform_distrib_terminal[1], load_uniform_distrib_
terminal[2], STREAM_LOADING), EVENT_LOADING);
    }
    else
    {
      if ((sim_time - bus_last_arrival) < (bus_min_time_process / 60))
      {
        event_schedule(bus_last_arrival + bus_min_time_process / 60, EVENT_BUS_DEPARTURE);
      }
      else
      {
        event_schedule(sim_time, EVENT_BUS_DEPARTURE);
      }
    }
  }
}

void load(void)
{
  if (transfer[3] == TERMINAL_1)
  {
    list_remove(FIRST, LIST_TERMINAL_1);
    person_first_arrival = transfer[1];
    timest(list_size[LIST_TERMINAL_1], VARIABLE_TERMINAL_1);
    sampst(sim_time - transfer[1], VARIABLE_DELAY_TERMINAL_1);
    transfer[1] = person_first_arrival;
    list_file(LAST, LIST_TO_CAR_RENTAL);
    timest(recent_bus_capacity(), VARIABLE_BUS);
    transfer[3] = TERMINAL_1;
    if (list_size[LIST_TERMINAL_1] > 0 && recent_bus_capacity() < MAX_BUS_CAPACITY)
    {
      event_schedule(sim_time + uniform(load_uniform_distrib_terminal[1], load_uniform_distrib_
terminal[2], STREAM_LOADING), EVENT_LOADING);
    }
    else
    {
      if ((sim_time - bus_last_arrival) < (bus_min_time_process / 60))
      {
        event_schedule(bus_last_arrival + bus_min_time_process / 60, EVENT_BUS_DEPARTURE);
      }
      else
      {
        event_schedule(sim_time, EVENT_BUS_DEPARTURE);
      }
    }
  }
  else if (transfer[3] == TERMINAL_2)
```

```
  {
    list_remove(FIRST, LIST_TERMINAL_2);
    person_first_arrival = transfer[1];
    timest(list_size[LIST_TERMINAL_2], VARIABLE_TERMINAL_2);
    sampst(sim_time - transfer[1], VARIABLE_DELAY_TERMINAL_2);
    transfer[1] = person_first_arrival;
    list_file(LAST, LIST_TO_CAR_RENTAL);
    timest(recent_bus_capacity(), VARIABLE_BUS);
    transfer[3] = TERMINAL_2;
    if (list_size[LIST_TERMINAL_2] > 0 && recent_bus_capacity() < MAX_BUS_CAPACITY)
    {
      event_schedule(sim_time + uniform(load_uniform_distrib_terminal[1], load_uniform_distrib_
terminal[2], STREAM_LOADING), EVENT_LOADING);
    }
    else
    {
      if ((sim_time - bus_last_arrival) < (bus_min_time_process / 60))
      {
        event_schedule(bus_last_arrival + bus_min_time_process / 60, EVENT_BUS_DEPARTURE);
      }
      else
      {
        event_schedule(sim_time, EVENT_BUS_DEPARTURE);
      }
    }
  }
  else
  {
    list_remove(FIRST, LIST_CAR_RENTAL);
    int destination = transfer[3];
    person_first_arrival = transfer[1];
    timest(list_size[LIST_CAR_RENTAL], VARIABLE_CAR_RENTAL);
    sampst(sim_time - transfer[1], VARIABLE_DELAY_CAR_RENTAL);
    transfer[1] = person_first_arrival;
    if (destination == 1)
    {
      list_file(LAST, LIST_TO_TERMINAL_1);
    }
    else
    {
      list_file(LAST, LIST_TO_TERMINAL_2);
    }
    timest(recent_bus_capacity(), VARIABLE_BUS);
    transfer[3] = CAR_RENTAL;
    if (list_size[LIST_CAR_RENTAL] > 0 && recent_bus_capacity() < MAX_BUS_CAPACITY)
    {
      event_schedule(sim_time + uniform(load_uniform_distrib_terminal[1], load_uniform_distrib_
terminal[2], STREAM_LOADING), EVENT_LOADING);
```

```
    }
    else
    {
      if ((sim_time - bus_last_arrival) < (bus_min_time_process / 60))
      {
        event_schedule(bus_last_arrival + bus_min_time_process / 60, EVENT_BUS_DEPARTURE);
      }
      else
      {
        event_schedule(sim_time, EVENT_BUS_DEPARTURE);
      }
    }
  }
}

void report(void)
{
  fprintf(outfile, "\n\n----------------------------STATISTIC REPORT IN HOUR-------------------
---------\n\n");
  fprintf(outfile, "a. Average and maximum number in each queue\n");
  fprintf(outfile, "Location          ");
  fprintf(outfile, "Average number         ");
  fprintf(outfile, "Maximum number\n");
  for (int i = VARIABLE_TERMINAL_1; i <= VARIABLE_CAR_RENTAL; i++)
  {
    timest(0.0, -i);
    if (i < MAX_NUM_LOCATION)
    {
      fprintf(outfile, "Terminal %d%17.3f%23.3f\n", i, transfer[1], transfer[2]);
    }
    else
    {
      fprintf(outfile, "Car rental %16.3f%23.3f\n", transfer[1], transfer[2]);
    }
  }
  fprintf(outfile, "\nb. Average and maximum delay in each queue\n");
  fprintf(outfile, "Location          ");
  fprintf(outfile, "Average delay          ");
  fprintf(outfile, "Maximum delay\n");
  int j = 1;
  for (int i = VARIABLE_DELAY_TERMINAL_1; i <= VARIABLE_DELAY_CAR_RENTAL; i++)
  {
    sampst(0.0, -i);
    if (i < VARIABLE_DELAY_CAR_RENTAL)
    {
      fprintf(outfile, "Terminal %d%17.3f%23.3f\n", j, transfer[1], transfer[3]);
    }
    else
```

```
    {
      fprintf(outfile, "Car rental %16.3f%23.3f\n", transfer[1], transfer[3]);
    }
    j++;
  }
  fprintf(outfile, "\nc. Average and maximum number on the bus\n");
  fprintf(outfile, "Average number    ");
  fprintf(outfile, "Maximum number\n");
  timest(0.0, -VARIABLE_BUS);
  fprintf(outfile, "%.3f%22.3f\n", transfer[1], transfer[2]);
  fprintf(outfile, "\nd. Average, maximum, and minimum time the bus stopped in each location\n"
);
  fprintf(outfile, "Location          ");
  fprintf(outfile, "Average time           ");
  fprintf(outfile, "Maximum time          ");
  fprintf(outfile, "Minimum time\n");
  j = 1;
  for (int i = VARIABLE_BUS_STOP_TERMINAL_1; i <= VARIABLE_BUS_STOP_CAR_RENTAL; i++)
  {
    sampst(0.0, -i);
    if (i < VARIABLE_BUS_STOP_CAR_RENTAL)
    {
      fprintf(outfile, "Terminal %d%17.3f%23.3f%21.3f\n", j, transfer[1], transfer[3], transfer
[4]);
    }
    else
    {
      fprintf(outfile, "Car rental %16.3f%23.3f%21.3f\n", transfer[1], transfer[3], transfer[4]
);
    }
    j++;
  }
  fprintf(outfile, "\ne. Average, maximum, and minimum the bus to make a loop\n");
  fprintf(outfile, "Average time      ");
  fprintf(outfile, "Maximum time           ");
  fprintf(outfile, "Minimum time\n");
  sampst(0.0, -VARIABLE_BUS_LOOP);
  fprintf(outfile, "%.3f%22.3f%23.3f\n", transfer[1], transfer[3], transfer[4]);
  fprintf(outfile, "\nf. Average, maximum, and minimum time person is in the system\n");
  fprintf(outfile, "Average time      ");
  fprintf(outfile, "Maximum time           ");
  fprintf(outfile, "Minimum time\n");
  sampst(0.0, -VARIABLE_PERSON);
  fprintf(outfile, "%.3f%22.3f%23.3f\n", transfer[1], transfer[3], transfer[4]);
}

void log_event()
{
```

```
  switch (next_event_type)
  {
  case EVENT_PERSON_ARRIVAL:
    printf("%.2lf EVENT_ARRIVAL_PERSON %f\n", sim_time, transfer[3]);
    break;
  case EVENT_BUS_ARRIVAL:
    printf("%.2lf EVENT_ARRIVAL_BUS %f\n", sim_time, transfer[3]);
    break;
  case EVENT_BUS_DEPARTURE:
    printf("%.2lf EVENT_DEPARTURE_BUS %f\n", sim_time, transfer[3]);
    break;
  case EVENT_UNLOADING:
    printf("%.2lf EVENT_UNLOADING %f\n", sim_time, transfer[3]);
    break;
  case EVENT_LOADING:
    printf("%.2lf EVENT_LOADING %f\n", sim_time, transfer[3]);
    break;
  }
}

int main()
{
  infile = fopen("car_rental.in", "r");
  outfile = fopen("car_rental.out", "w");

  fscanf(infile, "%d %d %lg", &num_location, &num_terminal, &length_simulation);
  for (int i = 1; i <= num_location; i++)
  {
    fscanf(infile, "%lg", &interarrival_time[i]);
  }
  fscanf(infile, "%lg %d", &bus_speed, &bus_start_location);
  for (int i = 1; i <= num_terminal; i++)
  {
    fscanf(infile, "%lg", &prob_distrib_terminal[i]);
  }
  for (int i = 1; i <= MAX_RANGE; i++)
  {
    fscanf(infile, "%lg", &unload_uniform_distrib_terminal[i]);
  }
  for (int i = 1; i <= MAX_RANGE; i++)
  {
    fscanf(infile, "%lg", &load_uniform_distrib_terminal[i]);
  }
  for (int i = 1; i <= MAX_NUM_LOCATION; i++)
  {
    for (int j = 1; j <= MAX_NUM_LOCATION; j++)
    {
      fscanf(infile, "%lg", &route_distance[i][j]);
```

```
    }
  }
  fscanf(infile, "%lg", &bus_min_time_process);

  fprintf(outfile, "Car Rental Model\n\n");
  fprintf(outfile, "Number of location%31d\n\n", num_location);
  fprintf(outfile, "Number of terminal%31d\n\n", num_terminal);
  fprintf(outfile, "Route distance from 1 to 2%25.1f mile\n\n", route_distance[1][2]);
  fprintf(outfile, "Route distance from 2 to 3%25.1f miles\n\n", route_distance[2][3]);
  fprintf(outfile, "Route distance from 3 to 1%25.1f miles\n\n", route_distance[3][1]);
  fprintf(outfile, "Bus speed%43.1f miles per hour\n\n", bus_speed);
  fprintf(outfile, "Minimum time spent by bus%26.1f minutes\n\n", bus_min_time_process);
  fprintf(outfile, "Distribution function of unloading          ");
  for (int i = 1; i <= num_terminal; i++)
  {
    i == 2 ? fprintf(outfile, "%7.1f", prob_distrib_terminal[i]) : fprintf(outfile, "%9.3f", pr
ob_distrib_terminal[i]);
  }
  fprintf(outfile, "\n");
  fprintf(outfile, "in each terminal\n\n");
  fprintf(outfile, "Interarrival rate per hour on each location");
  for (int i = 1; i <= num_location; i++)
  {
    i == 1 ? fprintf(outfile, "%9.1f", interarrival_time[i]) : fprintf(outfile, "%9.1f", intera
rrival_time[i]);
  }
  fprintf(outfile, "\n\n");
  fprintf(outfile, "Uniform distribution of unloading time range");
  for (int i = 1; i <= MAX_RANGE; i++)
  {
    i == 1 ? fprintf(outfile, "%8.1f - ", unload_uniform_distrib_terminal[i]) : fprintf(outfile
, "%.1f seconds each person", unload_uniform_distrib_terminal[i]);
  }
  fprintf(outfile, "\n\n");
  fprintf(outfile, "Uniform distribution of loading time range  ");
  for (int i = 1; i <= MAX_RANGE; i++)
  {
    i == 1 ? fprintf(outfile, "%8.1f - ", load_uniform_distrib_terminal[i]) : fprintf(outfile,
"%.1f seconds each person", load_uniform_distrib_terminal[i]);
  }
  fprintf(outfile, "\n\n");
  fprintf(outfile, "Length of simulation%32.1f hours", length_simulation);

  init_simlib();

  for (int i = 1; i <= MAX_RANGE; i++)
  {
    unload_uniform_distrib_terminal[i] = unload_uniform_distrib_terminal[i] / 3600;
```

```
    load_uniform_distrib_terminal[i] = load_uniform_distrib_terminal[i] / 3600;
  }

  for (int i = 1; i <= num_location; i++)
  {
    transfer[3] = i;
    event_schedule(expon(1 / interarrival_time[i], i), EVENT_PERSON_ARRIVAL);
  }
  transfer[3] = TERMINAL_1;
  event_schedule(route_distance[CAR_RENTAL][TERMINAL_1] / bus_speed, EVENT_BUS_ARRIVAL);
  event_schedule(length_simulation, EVENT_END_SIMULATION);

  do
  {
    timing();
    switch (next_event_type)
    {
    case EVENT_PERSON_ARRIVAL:
      person_arrive();
      break;
    case EVENT_BUS_ARRIVAL:
      bus_arrive();
      break;
    case EVENT_BUS_DEPARTURE:
      bus_depart();
      break;
    case EVENT_UNLOADING:
      unload();
      break;
    case EVENT_LOADING:
      load();
      break;
    case EVENT_END_SIMULATION:
      report();
      break;
    }
  } while (next_event_type != EVENT_END_SIMULATION);

  fclose(infile);
  fclose(outfile);

  return 0;
}
```

III. Output Program

Car Rental Model

Number of location	3
Number of terminal	2
Route distance from 1 to 2	1.0 mile
Route distance from 2 to 3	4.5 miles
Route distance from 3 to 1	4.5 miles
Bus speed	30.0 miles per hour
Minimum time spent by bus	5.0 minutes
Distribution function of unloading in each terminal	0.583 1.0
Interarrival rate per hour on each location	14.0 10.0 24.0
Uniform distribution of unloading time range	16.0 - 24.0 seconds each person
Uniform distribution of loading time range	15.0 - 25.0 seconds each person
Length of simulation	80.0 hours

----------STATISTIC REPORT IN HOUR----------

a. Average and maximum number in each queue

Location	Average number	Maximum number
Terminal 1	7.588	27.000
Terminal 2	4.927	17.000
Car rental	9.387	30.000

b. Average and maximum delay in each queue

Location	Average delay	Maximum delay
Terminal 1	0.539	1.633
Terminal 2	0.501	2.214
Car rental	0.405	1.123

c. Average and maximum number on the bus

Average number	Maximum number
13.855	20.000

d. Average, maximum, and minimum time the bus stopped in each location

Location	Average time	Maximum time	Minimum time
Terminal 1	0.112	0.170	0.083
Terminal 2	0.091	0.155	0.083
Car rental	0.190	0.229	0.098

e. Average, maximum, and minimum the bus to make a loop

Average time	Maximum time	Minimum time
0.726	0.793	0.600

f. Average, maximum, and minimum time person is in the system

Average time	Maximum time	Minimum time
0.761	2.476	0.183