**LAPORAN TUGAS BESAR I**

**IF4021 PEMODELAN DAN SIMULASI**



Oleh:

  Doddy Aditya Wiranugraha   13517008

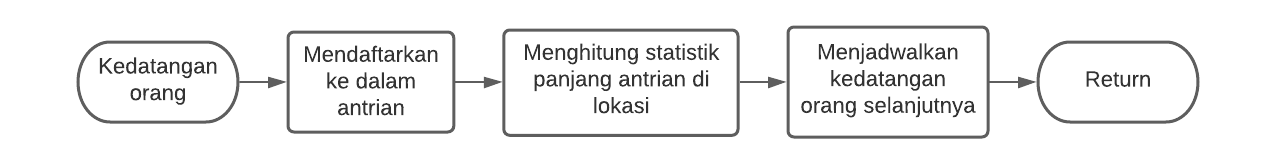
**PROGRAM STUDI TEKNIK INFORMATIKA**

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**

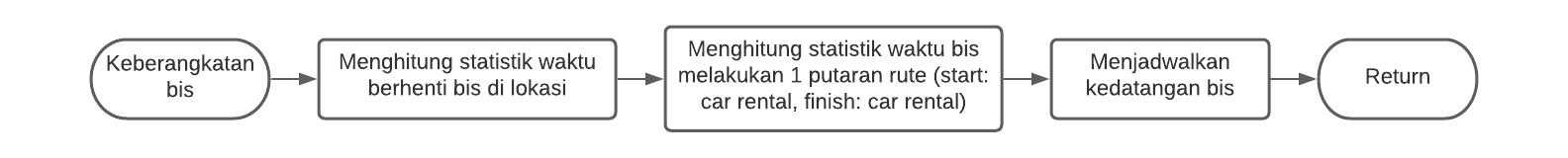
**INSTITUT TEKNOLOGI BANDUNG**

**2021**

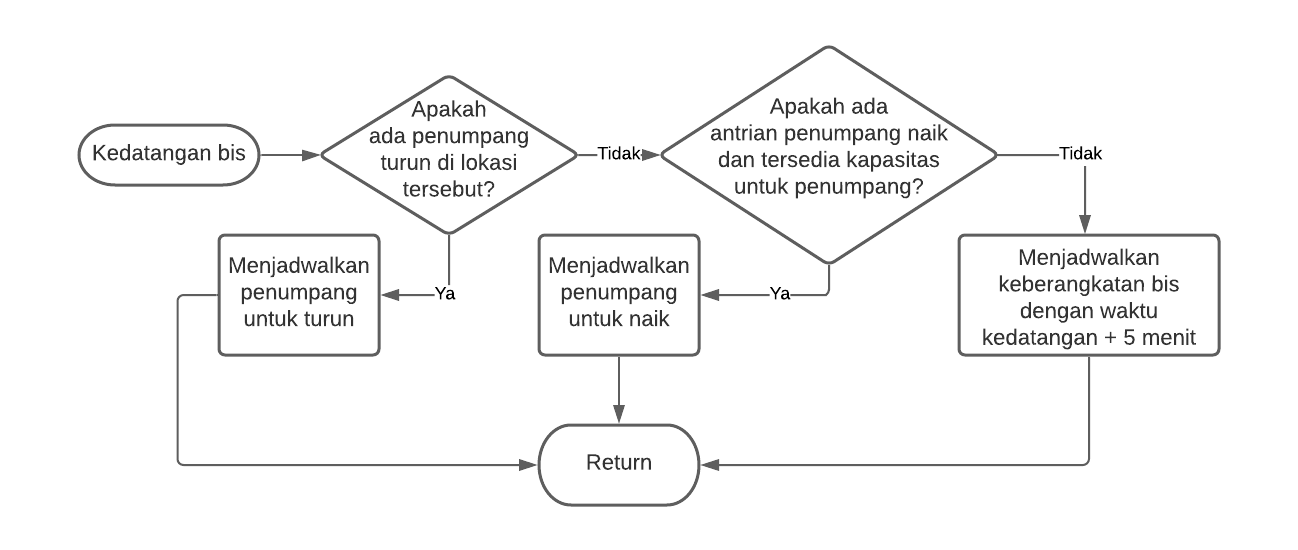
1. **Deskripsi Persoalan dan Solusi**
2. Event Kedatangan Orang



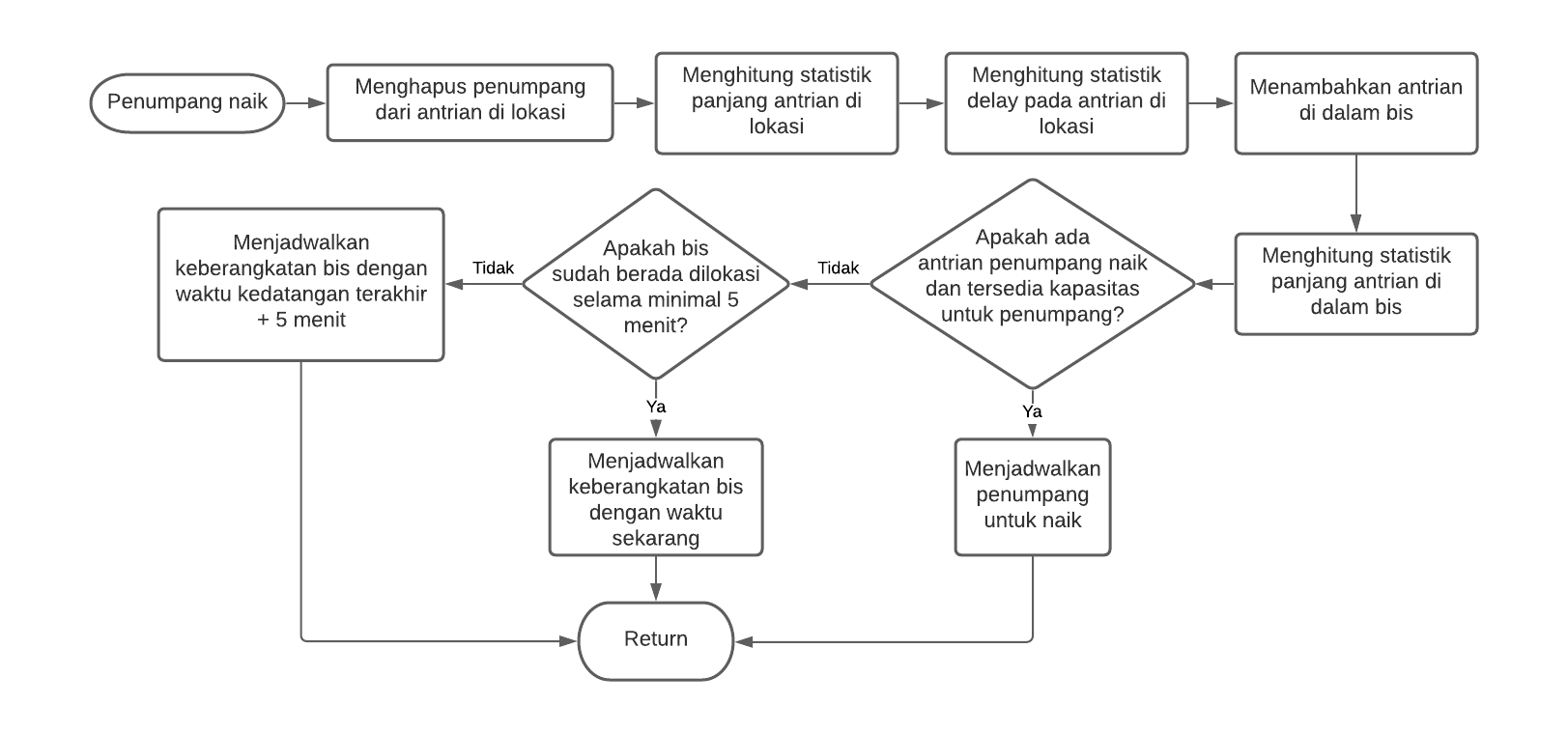
1. Event Keberangkatan Bis



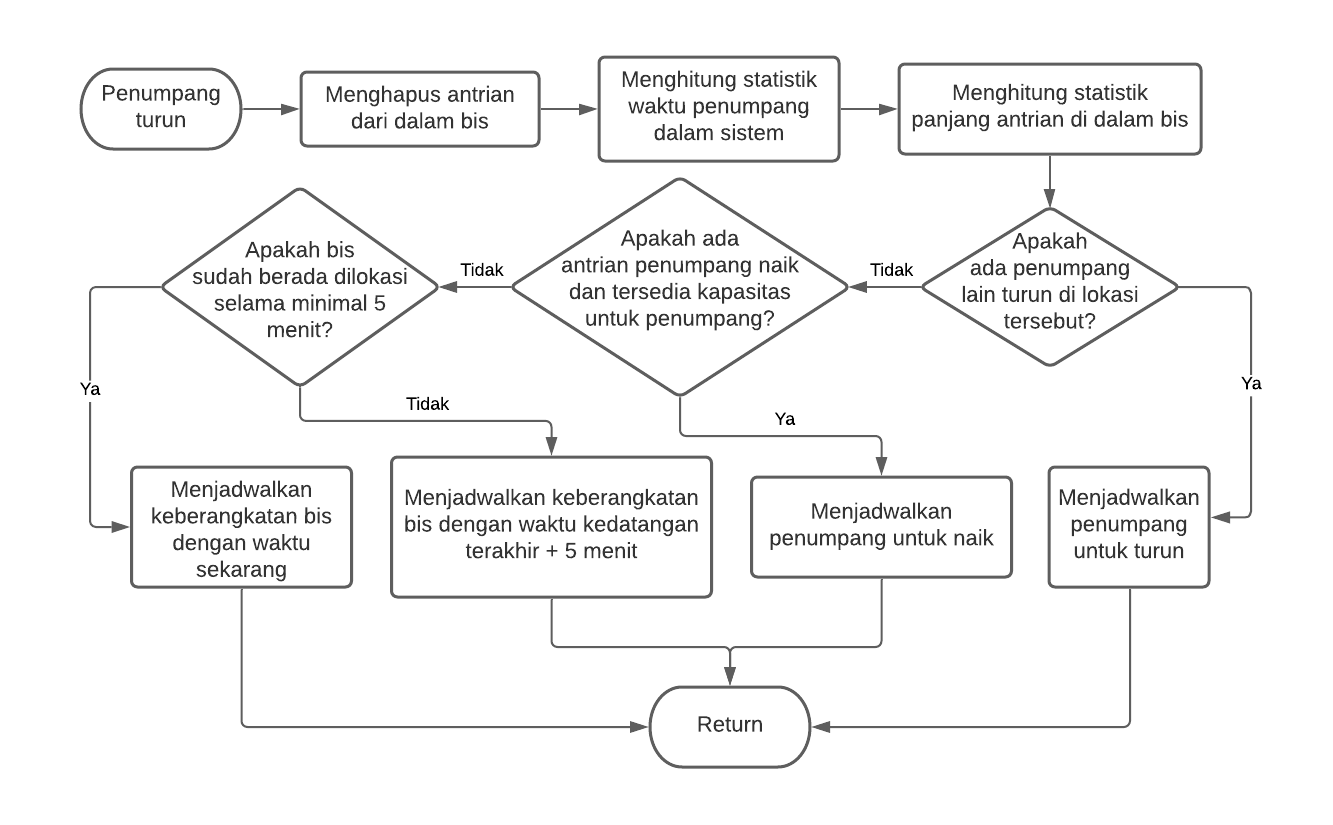
1. Event Kedatangan Bis



1. Event Penumpang Menaiki Bis



1. Event Penumpang Turun dari Bis



1. **Source Code**
2. 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 |

1. car\_rental.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\_distrib\_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\_distrib\_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\_distrib\_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\_ARRIVAL);    }    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\_ARRIVAL);    }    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\_ARRIVAL);    }  }  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\_distrib\_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\_distrib\_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\_distrib\_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", prob\_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", interarrival\_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;  } |

1. **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 0.583 1.0  in each terminal  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 |