

Tugas 1 IF4021 Inteligensi Buatan

Semester I Tahun 2021/2022

Simulasi Job Shop



Disusun oleh:

Gde Anantha Priharsena 13519026

Reihan Andhika Putra 13519043

Reyhan Emyr Arrosyid 13519167

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2021

Daftar Isi

Daftar Isi	2
Deskripsi Program	3
Kode Program	5
jobshop.c	5
jobshop.h	10
Hasil Run	12
Contoh file input (jobshop.in)	12
Contoh file output (jobshop.out)	12

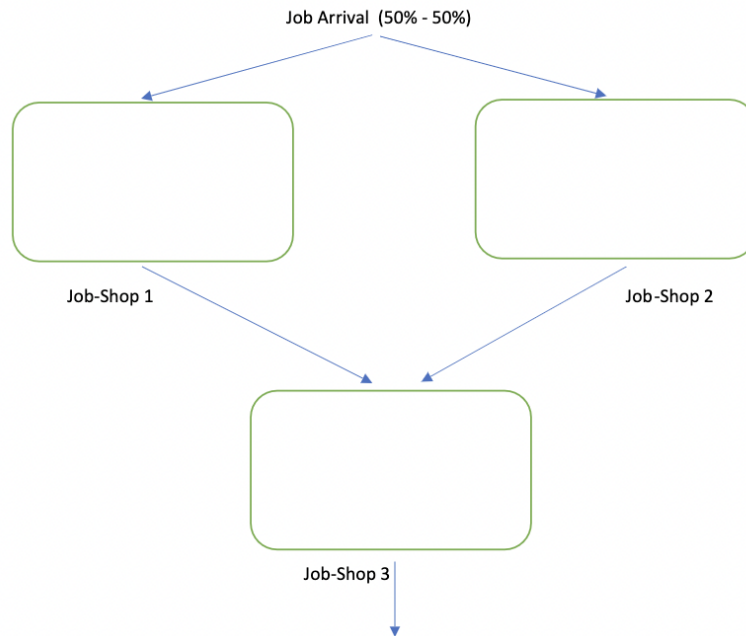
A. Deskripsi Program

Pertama-tama, program akan membaca file input bernama “jobshop.in”. File input ini berisi data awal seperti jumlah *station*, jumlah mesin di tiap *station*, jumlah *job type*, jumlah *task* setiap *job type*, fungsi distribusi pada setiap *job type*, rata-rata kedatangan orang ke *station*, dan panjang simulasi yang dibutuhkan untuk menjalankan program. Setelah itu, program akan memulai proses simulasi dan menginisialisasi seluruh mesin pada seluruh *station* yang ada dengan keadaan *idle*. Program juga akan menginisialisasi library simlib untuk membantu proses simulasi.

Untuk memulai proses simulasi, akan dilakukan proses randomisasi terhadap *event* pertama yang terjadi apakah seseorang akan datang ke jobshop 1 atau ke jobshop 2 dengan probabilitas 50:50. Setelah itu, program juga akan menjadwalkan akhir dari proses simulasi. Sehingga, setelah *event* pertama ditentukan, simlib akan melakukan proses simulasi hingga mencapai akhir dari proses simulasi tersebut.

Selama proses simulasi berlangsung, terdapat dua jenis *event* untuk setiap jobshop (6 *event* secara total) yaitu “*Arrival*” dan “*Departure*”. Event “*Arrival*” menyatakan terdapat seseorang yang datang dengan *job* pada sebuah *job shop* atau terdapat seseorang yang sedang singgah pada sebuah *jobshop/station* untuk mengunjungi *jobshop/station* lainnya. Sementara, *event* “*Departure*” menyatakan terdapat *job* yang meninggalkan sebuah *station*.

Berikut ini merupakan skema kedatangan orang pada setiap *jobshop* yang dilakukan selama simulasi.



Gambar 1. Skema Kedatangan Orang pada Setiap Jobshop

B. Kode Program

a. jobshop.c

```
/* External definitions for job-shop model. */

/* Required for use of simlib.c. */
#include "simlib.h"

/* Header for jobshop.c*/
#include "jobshop.h"

/* Declare non-simlib global variables. */

/* Number of station */
int num_stations;
/* Number of job types */
int num_job_types;
/* Loop index */
int i;
/* Loop index */
int j;
/* Number of machines in each station */
int num_machines[MAX_NUM_STATIONS + 1];
/* Number of tasks in each job type */
int num_tasks[MAX_NUM_JOB_TYPES + 1];
/* Route matrix */
int route[MAX_NUM_JOB_TYPES + 1][MAX_NUM_STATIONS + 1];
/* Number of used machine in each station */
int num_machines_busy[NUM_JOBSHOP+1][MAX_NUM_STATIONS + 1];
/* Type of the job*/
int job_type;
/* Task of the job (one job can have multiple task)*/
int task;
/*Current jobshop number*/
int jobshop_number;
/* Previous JobShop Number */
int previous_jobshop_number;
/* Previous Job Type*/
int previous_job_type = -1;

/* Mean interarrival time of each job */
double mean_interarrival;
/* Length of simulation (in hour)*/
double length_simulation;
/* Probability distribution for each job type */
double prob_distrib_job_type[26];
/* Mean service time for each job type and station */
double mean_service[MAX_NUM_JOB_TYPES + 1][MAX_NUM_STATIONS + 1];
/*Probability distribution of arriving in JobShop 1 or 2*/
/*The probability is equal*/
double prob_distrib_job_shop[3] = {0, 0.5, 1.0};

/* File to read input */
FILE *infile;
/* File to write output */
FILE *outfile;
```

```

void arrive (int new_job, int arrival_jopshop_number) {
    /* Function to serve as both an arrival event of a job to the system, as well as the non-event
    of a job's arriving to a subsequent station along its route. */

    int station, station_queue;

    /* If this is a new arrival to the system, generate the time of the next arrival and determine
    the job type and task number of the arriving job. */
    if (new_job == 1) {
        jobshop_number = random_integer (prob_distrib_job_shop, STREAM_JOBSHOP_ARRIVAL);
        event_schedule (sim_time + expon (mean_interarrival, STREAM_INTERARRIVAL),
            getArrivalJobShop(jobshop_number));
        // TODO: Remove old code.
        // event_schedule (sim_time + expon (mean_interarrival, STREAM_INTERARRIVAL), EVENT_ARRIVAL);
        job_type = random_integer (prob_distrib_job_type, STREAM_JOB_TYPE);
        task = 1;
    }

    if (arrival_jopshop_number == 3 && previous_job_type != -1) {
        job_type = previous_job_type;
        task = 1;
    }

    /* Determine the station from the route matrix. */
    station = route[job_type][task];
    station_queue = getListQueue(arrival_jopshop_number, station);

    /* LOG to file if the person just arrived in current jobshop */
    if (task == 1) {
        if (arrival_jopshop_number == 3) {
            fprintf (outfile, "[ %.2lf ] A Person just arrived in Jobshop %d, the job type are %d,
            previously from JobShop %d \n",
                sim_time, arrival_jopshop_number, job_type, previous_jobshop_number);
            previous_job_type = -1;
        } else {
            fprintf (outfile, "[ %.2lf ] A Person just arrived in Jobshop %d, the job type are %d \n",
                sim_time, arrival_jopshop_number, job_type);
        }
    }

    /* Check to see whether all machines in this station are busy. */
    if (num_machines_busy[arrival_jopshop_number][station] == num_machines[station]) {
        /* All machines in this station are busy, so place the arriving job at the end of the
        appropriate queue. Note that the following data are stored in the record for each job:
        1. Time of arrival to this station.
        2. Job type.
        3. Current task number. */

        transfer[1] = sim_time;
        transfer[2] = job_type;
        transfer[3] = task;
        list_file (LAST, station_queue);

        fprintf (outfile, "[ %.2lf ] The machine in station %d, in jobshop %d are busy, the job type is
        %d. Adding person in queue %d (%d person in queue) \n",
            sim_time, station, arrival_jopshop_number, job_type, station_queue,
            list_size[station_queue]);
    } else {
        /* A machine in this station is idle, so start service on the arriving
        job (which has a delay of zero). */

        // TODO: Remove old code and evaluate our new code (sampst and timest).
        // sampst (0.0, station); /* For station. */
        // sampst (0.0, num_stations + job_type); /* For job type. */
        // timest ((double) num_machines_busy[station], station);

        sampst (0.0, station_queue); /* For station. */
        sampst (0.0, 3 * num_stations + job_type); /* For overall job type. */
        sampst (0.0, 3 * (num_stations + arrival_jopshop_number) + job_type); /* For job type of
        individual jobshop. */
        ++num_machines_busy[arrival_jopshop_number][station];
        timest ((double) num_machines_busy[arrival_jopshop_number][station], station_queue);

        /* Schedule a service completion. Note defining attributes beyond the
        first two for the event record before invoking event_schedule. */

        transfer[3] = job_type;
        transfer[4] = task;
        event_schedule (sim_time + erlang (2, mean_service[job_type][task],
            STREAM_SERVICE), getDepartureJobShop(arrival_jopshop_number));
    }
}

```

```

void depart (int departure_jobshop_number) {
    /* Event function for departure of a job from a particular station. */
    int station, job_type_queue, task_queue, station_queue;

    /* Determine the station from which the job is departing. */
    job_type = transfer[3];
    task = transfer[4];
    station = route[job_type][task];
    station_queue = getListQueue(departure_jobshop_number, station);

    /* Check to see whether the queue for this station is empty. */
    if (list_size[station_queue] == 0) {
        /* The queue for this station is empty, so make a machine in this
           station idle. */
        --num_machines_busy[departure_jobshop_number][station];
        // TODO: Remove old code and evaluate our new code (timest).
        // timest ((double) num_machines_busy[station], station);
        timest((double) num_machines_busy[departure_jobshop_number][station], station_queue);
    } else {
        /* The queue is nonempty, so start service on first job in queue. */
        list_remove (FIRST, station_queue);

        fprintf (outfile, "[%2lf ] Process person from queue %d. Resuming job in jobshop %d station %d
(%d person in queue) \n",
            sim_time, station_queue, departure_jobshop_number, station, list_size[station_queue]);

        // TODO: Remove old code and evaluate our new code (sampst).
        /* Tally this delay for this station. */
        // sampst (sim_time - transfer[1], station);
        sampst(sim_time - transfer[1], station_queue);

        /* Tally this same delay for this job type. */
        job_type_queue = transfer[2];
        task_queue = transfer[3];
        sampst (sim_time - transfer[1], 3 * num_stations + job_type_queue);
        sampst (sim_time - transfer[1], 3 * (num_stations + departure_jobshop_number) + job_type_queue);

        /* Schedule end of service for this job at this station. Note defining attributes beyond
           the first two for the event record before invoking event_schedule. */
        transfer[3] = job_type_queue;
        transfer[4] = task_queue;
        event_schedule (sim_time + erlang (2, mean_service[job_type_queue][task_queue],
            STREAM_SERVICE), getDepartureJobShop(departure_jobshop_number));
    }

    /* If the current departing job has one or more tasks yet to be done, send the job to the next
       station on its route. */
    if (task < num_tasks[job_type]) {
        ++task;
        arrive (0, departure_jobshop_number);
    } else if (task == num_tasks[job_type]) {
        /* Determine if the person will depart from current JobShop */
        fprintf (outfile, "[%2lf ] A Person just departed from Jobshop %d, the job type are %d \n",
            sim_time, departure_jobshop_number, job_type);
        /* If the current person has no more task and not from jobshop 3 then assign it to jobshop 3 */
        if (departure_jobshop_number != 3) {
            int *temp;
            // temp = &previous_job_type;
            // *temp = job_type;
            previous_job_type = job_type;
            previous_jobshop_number = departure_jobshop_number;
            event_schedule (sim_time, getArrivalJobShop(3));
        }
    }
}
}

```

```

void report (void) {
    // TODO: Check the correctness of this function.
    /* Report generator function. */

    int station_queue;
    double overall_avg_job_tot_delay, avg_job_tot_delay, sum_probs;

    /* Compute the average total delay in queue for each job type and the overall average job total
       delay. */
    fprintf (outfile, "\n\nOverall statistics for jobshop system");
    fprintf (outfile, "\nJob type      Average total delay in queue");
    overall_avg_job_tot_delay = 0.0;
    sum_probs = 0.0;
    for (i = 1; i <= num_job_types; ++i) {
        avg_job_tot_delay = sampst (0.0, -(3 * num_stations + i)) * num_tasks[i];
        fprintf (outfile, "\n\n%4d%27.3f", i, avg_job_tot_delay);
        overall_avg_job_tot_delay += (prob_distrib_job_type[i] - sum_probs) * avg_job_tot_delay;
        sum_probs = prob_distrib_job_type[i];
    }

    fprintf (outfile, "\n\nOverall average job total delay =%10.3f", overall_avg_job_tot_delay);
}

```

```

for (i = 1; i <= 3; ++i) {
    /* Compute the average number in queue, the average utilization, and the
       average delay in queue for each station for each jobshop. */
    fprintf (outfile, "\n\nStatistics for jobshop %d", i);
    fprintf (outfile, "\n\nJob type      Average total delay in queue");

    overall_avg_job_tot_delay = 0.0;
    sum_probs = 0.0;
    for (j = 1; j <= num_job_types; ++j) {
        avg_job_tot_delay = sampst (0.0, -(3 * (num_stations + i) + j)) * num_tasks[j];
        fprintf (outfile, "\n\n%4d%27.3f", j, avg_job_tot_delay);
        overall_avg_job_tot_delay += (prob_distrib_job_type[j] - sum_probs) * avg_job_tot_delay;
        sum_probs = prob_distrib_job_type[j];
    }

    fprintf (outfile, "\n\nOverall average job total delay =%10.3f", overall_avg_job_tot_delay);

    fprintf (outfile, "\n\n Work      Average number      Average      Average delay");
    fprintf (outfile, "\n\n station      in queue      utilization      in queue");
    for (j = 1; j <= num_stations; ++j) {
        station_queue = getListQueue(i, j);
        fprintf (outfile, "\n\n%4d%17.3f%17.3f%17.3f", j, filest (station_queue,
            timest (0.0, -station_queue) / num_machines[j], sampst (0.0, -station_queue)));
    }
}

}

void readJobShopParameter(void) {
    /* Open input and output files. */
    infile = fopen ("jobshop.in", "r");
    outfile = fopen ("jobshop.out", "w");

    /* Read input parameters. */
    /* Read basic data, such as number of station, jobtype, mean interarrival and length simulation */
    fscanf (infile, "%d %d %lg %lg", &num_stations, &num_job_types, &mean_interarrival,
        &length_simulation);

    /* Read the number of machine in each station. */
    for (j = 1; j <= num_stations; ++j){
        fscanf (infile, "%d", &num_machines[j]);
    }

    /* Read the number of tasks to be done in each job type. */
    for (i = 1; i <= num_job_types; ++i){
        fscanf (infile, "%d", &num_tasks[i]);
    }

    /* Foreach job type, read it's job task attribute */
    for (i = 1; i <= num_job_types; ++i) {
        /* Read the job station */
        for (j = 1; j <= num_tasks[i]; ++j) {
            fscanf (infile, "%d", &route[i][j]);
        }
        /* Read the job service time for each station */
        for (j = 1; j <= num_tasks[i]; ++j) {
            fscanf (infile, "%lg", &mean_service[i][j]);
        }
    }

    /* Read the probability distribution of job type. */
    for (i = 1; i <= num_job_types; ++i) {
        fscanf (infile, "%lg", &prob_distrib_job_type[i]);
    }

    /* Write report heading and input parameters. */
    fprintf (outfile, "Job-shop model\n\n");
    fprintf (outfile, "Number of work stations%21d\n\n", num_stations);
    fprintf (outfile, "Number of machines in each station      ");
    for (j = 1; j <= num_stations; ++j) {
        fprintf (outfile, "%5d", num_machines[j]);
    }
    fprintf (outfile, "\n\nNumber of job types%25d\n\n", num_job_types);
    fprintf (outfile, "Number of tasks for each job type      ");
    for (i = 1; i <= num_job_types; ++i) {
        fprintf (outfile, "%5d", num_tasks[i]);
    }
    fprintf (outfile, "\n\nDistribution function of job types  ");
    for (i = 1; i <= num_job_types; ++i) {
        fprintf (outfile, "%8.3f", prob_distrib_job_type[i]);
    }
    fprintf (outfile, "\n\nMean interarrival time of jobs%14.2f hours\n\n", mean_interarrival);
    fprintf (outfile, "Length of the simulation%20.1f eight-hour days\n\n", length_simulation);
    fprintf (outfile, "Job type      Work stations on route");
    for (i = 1; i <= num_job_types; ++i) {
        fprintf (outfile, "\n\n%4d      ", i);
        for (j = 1; j <= num_tasks[i]; ++j) {
            fprintf (outfile, "%5d", route[i][j]);
        }
    }
    fprintf (outfile, "\n\n\nJob type      ");
    fprintf (outfile, "Mean service time (in hours) for successive tasks");
    for (i = 1; i <= num_job_types; ++i){
        fprintf (outfile, "\n\n%4d      ", i);
        for (j = 1; j <= num_tasks[i]; ++j) {
            fprintf (outfile, "%9.2f", mean_service[i][j]);
        }
    }
    fprintf (outfile, "\n\nLOG \n");
}

```



```

void init () {
    /* Initiate jobshop problem. */
    printf("Running Job Shop scheduling\n");

    /* Initialize all machines in all stations to the idle state. */
    for (i=1; i <= NUM_JOBSHOP; i++) {
        for (j = 1; j <= num_stations; ++j) {
            num_machines_busy[i][j] = 0;
        }
    }

    /* Initialize simlib */
    init_simlib ();

    /* Set maxatr = max(maximum number of attributes per record, 4) */
    maxatr = 4; /* NEVER SET maxatr TO BE SMALLER THAN 4. */

    /* Schedule first arrival with probability 50:50 for jobshop 1 and 2 */
    jobshop_number = random_integer (prob_distrib_job_shop, STREAM_JOBSHOP_ARRIVAL);
    event_schedule (expon (mean_interarrival, STREAM_INTERARRIVAL), getArrivalJobShop(jobshop_number));

    // TODO: Remove old code.
    // /* Schedule the arrival of the first job. */
    // event_schedule (expon (mean_interarrival, STREAM_INTERARRIVAL), EVENT_ARRIVAL);

    /* Schedule the end of the simulation. (This is needed for consistency of
       units.) */
    event_schedule (8 * length_simulation, EVENT_END_SIMULATION);
}

int main () {
    /* Main function. */

    /* Read JobShop specification from the input file. */
    readJobShopParameter();

    /* Initialize the jobshop problem. */
    init();

    /* Run the simulation until it terminates after an end-simulation event
       (type EVENT_END_SIMULATION) occurs. */
    do {
        /* Determine the next event. */
        timing ();

        /* Invoke the appropriate event function. */
        switch (next_event_type) {
            // TODO: Remove old code
            // case EVENT_ARRIVAL:
            //     fprintf (outfile, "[ %.2lf ] A Person just arrived \n", sim_time);
            //     arrive (1);
            //     break;
            // case EVENT_DEPARTURE:
            //     fprintf (outfile, "[ %.2lf ] A Person just departed \n", sim_time);
            //     depart ();
            //     break;
            case EVENT_ARRIVAL_JOBSHOP_1:
                arrive(1, 1);
                break;
            case EVENT_ARRIVAL_JOBSHOP_2:
                arrive(1, 2);
                break;
            case EVENT_ARRIVAL_JOBSHOP_3:
                arrive(0, 3);
                break;
            case EVENT_DEPARTURE_JOBSHOP_1:
                depart(1);
                break;
            case EVENT_DEPARTURE_JOBSHOP_2:
                depart(2);
                break;
            case EVENT_DEPARTURE_JOBSHOP_3:
                depart(3);
                break;
            case EVENT_END_SIMULATION:
                fprintf (outfile, "[ %.2lf ] Simulation ended \n", sim_time);
                report ();
                break;
        }
        /* If the event just executed was not the end-simulation event (type EVENT_END_SIMULATION),
           continue simulating. Otherwise, end the simulation. */
    } while (next_event_type != EVENT_END_SIMULATION);

    fclose (infile);
    fclose (outfile);

    return 0;
}

```

b. jobshop.h

```

/*
 *
 *
 */
#ifndef __JOBSHOP_H__
#define __JOBSHOP_H__

/* Event detail */
/* Event type for arrival of a job to the system. */
#define EVENT_ARRIVAL 1
/* Event type for departure of a job from a particular station. */
#define EVENT_DEPARTURE 2
/* Event type for end of the simulation. */
#define EVENT_END_SIMULATION 3
/* Event type for Arrival in jobshop 1. */
#define EVENT_ARRIVAL_JOBSHOP_1 4
/* Event type for Departure in jobshop 1. */
#define EVENT_DEPARTURE_JOBSHOP_1 5
/* Event type for Arrival in jobshop 2. */
#define EVENT_ARRIVAL_JOBSHOP_2 6
/* Event type for Departure in jobshop 2. */
#define EVENT_DEPARTURE_JOBSHOP_2 7
/* Event type for Arrival in jobshop 3. */
#define EVENT_ARRIVAL_JOBSHOP_3 8
/* Event type for Departure in jobshop 3. */
#define EVENT_DEPARTURE_JOBSHOP_3 9

/* Jobshop specification */
/* Number of jobshop */
#define NUM_JOBSHOP 3
/* Maximum number of stations in each jobshop. */
#define MAX_NUM_STATIONS 5
/* Maximum number of job types. */
#define MAX_NUM_JOB_TYPES 3
/* Random-number stream for interarrivals. */
#define STREAM_INTERARRIVAL 1
/* Random-number stream for job types. */
#define STREAM_JOB_TYPE 2
/* Random-number stream for service times. */
#define STREAM_SERVICE 3
/* Random-number stream for jobshop arrival */
#define STREAM_JOBSHOP_ARRIVAL 2

/* Variable */
/* Number of station */
extern int num_stations;
/* Number of job types */
extern int num_job_types;
/* Loop index */
extern int i;
/* Loop index */
extern int j;
/* Number of machines in each station */
extern int num_machines[MAX_NUM_STATIONS + 1];
/* Number of tasks in each job type */
extern int num_tasks[MAX_NUM_JOB_TYPES + 1];
/* Route matrix */
extern int route[MAX_NUM_JOB_TYPES + 1][MAX_NUM_STATIONS + 1];
/* Number of used machine in each station */
extern int num_machines_busy[NUM_JOBSHOP + 1][MAX_NUM_STATIONS + 1], job_type,
task_type of the job*/
extern int job_type;
/* Task of the job (one job can have multiple task)*/
extern int task;
/* Current jobshop number */
extern int jobshop_number;
/* Previous JobShop Number */
extern int previous_jobshop_number;
/* Previous Job Type */
extern int previous_job_type;

```

```

/* Mean interarrival time of each job */
extern double mean_interarrival;
/* Length of simulation (in hour)*/
extern double length_simulation;
/* Probability distribution for each job type */
extern double prob_distrib_job_type[26];
/* Mean service time for each job type and station */
extern double mean_service[MAX_NUM_JOB_TYPES + 1][MAX_NUM_STATIONS + 1];
/*Probability distribution of arriving in JobShop 1 or 2*/
extern double prob_distrib_job_shop[3];

/* File to read input */
extern FILE *infile;
/* File to write output */
extern FILE *outfile;

/* Function */
/**
 * @brief Read JobShop specification from the input file. The specification includes the
 * number of job types, the number of tasks for each job type, the number of stations,
 * the number of machines at each station, the mean interarrival time, the mean service time for
 * each job type and task, and the probability distribution for each job type.
 */
void readJobShopParameter(void);

/**
 * @brief Initialize the JobShop simulation.
 */
void initJobShop(void);

/**
 * @brief Arrive a new job to the system or to new station.
 *
 * @param int new_job - 1 if a new job arrives, 0 if a new station arrives.
 * @param int arrival_jobshop_number - jobshop number.
 */
void arrive (int new_job, int arrival_jobshop_number);

/**
 * @brief Departure job from particular station.
 *
 * @param int departure_jobshop_number - jobshop number.
 */
void depart (int departure_jobshop_number);

/**
 * @brief Print the simulation result.
 */
void report (void);

/**
 * @brief Get event arrival Jobshop Number.
 *
 * @param int jobshop_number
 * @return int
 */
int getArrivalJobShop(int jobshop_number){
    switch (jobshop_number) {
        case 1:
            return EVENT_ARRIVAL_JOBSHOP_1;
        case 2:
            return EVENT_ARRIVAL_JOBSHOP_2;
        case 3:
            return EVENT_ARRIVAL_JOBSHOP_3;
    }
}

/**
 * @brief Get event departure Jobshop Number.
 *
 * @param int jobshop_number
 * @return int
 */
int getDepartureJobShop(int jobshop_number){
    switch (jobshop_number) {
        case 1:
            return EVENT_DEPARTURE_JOBSHOP_1;
        case 2:
            return EVENT_DEPARTURE_JOBSHOP_2;
        case 3:
            return EVENT_DEPARTURE_JOBSHOP_3;
    }
}

/**
 * @brief Get list queue of specific station in specific JobShop.
 *
 * @param int jobshop_number
 * @param int station_number
 * @return int
 */
int getListQueue(int jobshop_number, int station_number){
    // The rule is: listQueue = (jobshop_number - 1) * num_stations + station_number
    return (jobshop_number - 1) * num_stations + station_number;
}
#endif

```

C. Hasil Run

a. Contoh file input (jobshop.in)

5	3	0.25	10.0	
3	2	4	3	1
4	3	5		
3	1	2	5	
0.5	0.6	0.85	0.5	
4	1	3		
1.1	0.8	0.75		
2	5	1	4	3
1.2	0.25	0.7	0.9	1.0
0.3	0.8	1.0		

b. Contoh file output (jobshop.out)

Job-shop model					
Number of work stations	5				
Number of machines in each station	3	2	4	3	1
Number of job types	3				
Number of tasks for each job type	4	3	5		
Distribution function of job types	0.300	0.800	1.000		
Mean interarrival time of jobs	0.25 hours				
Length of the simulation	10.0 eight-hour days				
Job type	Work stations on route				
1	3	1	2	5	
2	4	1	3		
3	2	5	1	4	3
Job type	Mean service time (in hours) for successive tasks				
1	0.50	0.60	0.85	0.50	
2	1.10	0.80	0.75		
3	1.20	0.25	0.70	0.90	1.00
LOG					
[0.23] A Person just arrived in Jobshop 1, the job type are 2					
[0.35] A Person just arrived in Jobshop 1, the job type are 1					
[0.56] A Person just arrived in Jobshop 2, the job type are 3					
[0.80] A Person just arrived in Jobshop 2, the job type are 1					
[0.97] A Person just arrived in Jobshop 2, the job type are 3					
[1.06] A Person just arrived in Jobshop 1, the job type are 1					
[1.48] A Person just arrived in Jobshop 1, the job type are 1					
[1.52] A Person just arrived in Jobshop 2, the job type are 3					
[1.52] The machine in station 2, in jobshop 2 are busy, the job type is 3. Adding person in queue 7 (1 person in queue)					
[1.63] A Person just arrived in Jobshop 2, the job type are 1					
[1.68] Process person from queue 7. Resuming job in jobshop 2 station 2 (0 person in queue)					
[1.82] A Person just departed from Jobshop 1, the job type are 2					
[1.82] A Person just arrived in Jobshop 3, the job type are 2, previously from JobShop 1					
[1.88] A Person just arrived in Jobshop 1, the job type are 2					
[2.14] A Person just arrived in Jobshop 1, the job type are 3					
[2.14] The machine in station 2, in jobshop 1 are busy, the job type is 3. Adding person in queue 2 (1 person in queue)					
[2.21] A Person just arrived in Jobshop 1, the job type are 2					
[2.41] A Person just arrived in Jobshop 2, the job type are 1					
[2.42] The machine in station 2, in jobshop 1 are busy, the job type is 1. Adding person in queue 2 (2 person in queue)					
[2.43] A Person just arrived in Jobshop 1, the job type are 2					
[2.46] A Person just departed from Jobshop 2, the job type are 1					
[2.46] A Person just arrived in Jobshop 3, the job type are 1, previously from JobShop 2					
[2.46] Process person from queue 2. Resuming job in jobshop 1 station 2 (1 person in queue)					
[2.58] A Person just departed from Jobshop 1, the job type are 1					
[2.58] A Person just arrived in Jobshop 3, the job type are 1, previously from JobShop 1					
[2.59] Process person from queue 2. Resuming job in jobshop 1 station 2 (0 person in queue)					

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

```

[ 79.15 ] The machine in station 4, in jobshop 3 are busy, the job type is 2. Adding person in queue 14 (2 person in queue)
[ 79.21 ] Process person from queue 11. Resuming job in jobshop 3 station 1 (12 person in queue)
[ 79.21 ] The machine in station 3, in jobshop 3 are busy, the job type is 2. Adding person in queue 13 (3 person in queue)
[ 79.22 ] Process person from queue 13. Resuming job in jobshop 3 station 3 (2 person in queue)
[ 79.22 ] A Person just departed from Jobshop 3, the job type are 3
[ 79.23 ] Process person from queue 6. Resuming job in jobshop 2 station 1 (3 person in queue)
[ 79.23 ] Process person from queue 15. Resuming job in jobshop 3 station 5 (3 person in queue)
[ 79.23 ] A Person just departed from Jobshop 3, the job type are 1
[ 79.33 ] A Person just arrived in Jobshop 2, the job type are 1
[ 79.35 ] The machine in station 1, in jobshop 2 are busy, the job type is 2. Adding person in queue 6 (4 person in queue)
[ 79.42 ] Process person from queue 7. Resuming job in jobshop 2 station 2 (1 person in queue)
[ 79.44 ] Process person from queue 13. Resuming job in jobshop 3 station 3 (1 person in queue)
[ 79.44 ] A Person just departed from Jobshop 3, the job type are 2
[ 79.44 ] Process person from queue 11. Resuming job in jobshop 3 station 1 (11 person in queue)
[ 79.44 ] The machine in station 2, in jobshop 3 are busy, the job type is 1. Adding person in queue 12 (12 person in queue)
[ 79.46 ] A Person just arrived in Jobshop 1, the job type are 2
[ 79.47 ] The machine in station 1, in jobshop 2 are busy, the job type is 1. Adding person in queue 6 (5 person in queue)
[ 79.48 ] Process person from queue 13. Resuming job in jobshop 3 station 3 (0 person in queue)
[ 79.48 ] A Person just departed from Jobshop 3, the job type are 2
[ 79.49 ] Process person from queue 6. Resuming job in jobshop 2 station 1 (4 person in queue)
[ 79.57 ] A Person just arrived in Jobshop 1, the job type are 3
[ 79.63 ] Process person from queue 14. Resuming job in jobshop 3 station 4 (1 person in queue)
[ 79.63 ] The machine in station 3, in jobshop 3 are busy, the job type is 3. Adding person in queue 13 (1 person in queue)
[ 79.64 ] A Person just departed from Jobshop 2, the job type are 1
[ 79.64 ] A Person just arrived in Jobshop 3, the job type are 1, previously from JobShop 2
[ 79.64 ] The machine in station 3, in jobshop 3 are busy, the job type is 1. Adding person in queue 13 (2 person in queue)
[ 79.67 ] The machine in station 2, in jobshop 1 are busy, the job type is 1. Adding person in queue 2 (1 person in queue)
[ 79.68 ] A Person just arrived in Jobshop 1, the job type are 3
[ 79.68 ] The machine in station 2, in jobshop 1 are busy, the job type is 3. Adding person in queue 2 (2 person in queue)
[ 79.68 ] The machine in station 1, in jobshop 2 are busy, the job type is 1. Adding person in queue 6 (5 person in queue)
[ 79.68 ] Process person from queue 13. Resuming job in jobshop 3 station 3 (1 person in queue)
[ 79.68 ] The machine in station 1, in jobshop 3 are busy, the job type is 1. Adding person in queue 11 (12 person in queue)
[ 79.72 ] Process person from queue 11. Resuming job in jobshop 3 station 1 (11 person in queue)
[ 79.72 ] The machine in station 3, in jobshop 3 are busy, the job type is 2. Adding person in queue 13 (2 person in queue)
[ 79.78 ] Process person from queue 6. Resuming job in jobshop 2 station 1 (4 person in queue)
[ 79.78 ] The machine in station 2, in jobshop 2 are busy, the job type is 1. Adding person in queue 7 (2 person in queue)
[ 79.78 ] A Person just arrived in Jobshop 1, the job type are 1
[ 79.83 ] Process person from queue 12. Resuming job in jobshop 3 station 2 (11 person in queue)
[ 79.83 ] The machine in station 5, in jobshop 3 are busy, the job type is 1. Adding person in queue 15 (4 person in queue)
[ 79.83 ] Process person from queue 15. Resuming job in jobshop 3 station 5 (3 person in queue)
[ 79.83 ] A Person just departed from Jobshop 3, the job type are 1
[ 79.88 ] Process person from queue 6. Resuming job in jobshop 2 station 1 (3 person in queue)
[ 79.88 ] The machine in station 2, in jobshop 2 are busy, the job type is 1. Adding person in queue 7 (3 person in queue)
[ 79.90 ] Process person from queue 2. Resuming job in jobshop 1 station 2 (1 person in queue)
[ 79.95 ] Process person from queue 15. Resuming job in jobshop 3 station 5 (2 person in queue)
[ 79.95 ] A Person just departed from Jobshop 3, the job type are 1
[ 79.97 ] The machine in station 3, in jobshop 1 are busy, the job type is 2. Adding person in queue 3 (1 person in queue)
[ 80.00 ] Simulation ended

```

Overall statistics for jobshop system

Job type Average total delay in queue

1	2.523
2	1.700
3	3.328

Overall average job total delay = 2.273

Statistics for jobshop 1

Job type Average total delay in queue

1	0.497
2	0.062
3	0.526

Overall average job total delay = 0.285

Work station	Average number in queue	Average utilization	Average delay in queue
1	0.105	0.471	0.052
2	0.203	0.508	0.196
3	0.012	0.328	0.006
4	0.045	0.481	0.034
5	0.205	0.429	0.207

Statistics for jobshop 2

Job type Average total delay in queue

1	0.455
2	0.206
3	0.716

Overall average job total delay = 0.383

Work station	Average number in queue	Average utilization	Average delay in queue
1	0.219	0.472	0.092
2	0.206	0.479	0.175
3	0.024	0.370	0.011
4	0.203	0.517	0.139
5	0.215	0.450	0.202

Statistics for jobshop 3

Job type	Average total delay in queue
1	4.865
2	3.364
3	6.594

Overall average job total delay = 4.460

Work station	Average number in queue	Average utilization	Average delay in queue
1	3.415	0.836	0.849
2	4.426	0.835	2.182
3	0.624	0.702	0.165
4	5.705	0.928	2.201
5	2.924	0.783	1.673