Simulation

Exercise 1.26

System description, input date/distributions:
see description of the exercise

Assumptions:
no new assumptions are made

State:
server A_1 idle/busy1 (i.e. busy with type 1 customer) /busy2
server A_2 idle/busy1/busy2
server B idle/busy1/busy2
number of customers in queue 1 with their arrival times
number of customers in queue 2 with their arrival times.

Events:
arrival of customer
departure1A: departure of type 1 customer from server of type A
departure1B: departure of type 1 customer from server of type B
departure of type 2 customer

Performance measures:
average delay in queue for each type of customer:

Let D_1 be the total delay of type 1 customers in the simulation and n_1 be the number of type 1 customers for which delay has been measured. If a customer enters service, increase D_1 by the delay of this customer and increase n_1 by 1.At the end of the simulation the average delay equals D_1/n_1. Same is applied for customers of type 2.

average number of customers in queue 1 en queue 2.

Note that this is an average over time. Let Q_1 be the surface under the graph of the queue-length of queue 1 as a function of time until the current time. At each event we update Q_1 by adding *(time – $time_{previous\ event}$)*(queue length during this interval).* At the end the average number of customers in the queue equals Q_1 divided by the total time. Similarly, we have Q_2.

expected portion of time that each server spends on customer of type 1 and type 2

Let $T_{1,S}$ be the amount of time that server S ($S=A_1,A_2,B$) is busy until the current time. At each event update $T_{1,S}$ by adding *time – $time_{previous\ event}$* if the server S was busy working for customer 1 during the interval [$time_{previous\ event}$.,*time*]. Similarly, we have the number $T_{2,S}$. At the end, these numbers have to be the divided by the total time.

Pseudo-code:

```
 while time < runlength
{
case nextevent of
        arrival:
                {
                        schedule new arrival;
                        determine client type (with random generator);
                        update busy time of servers and total queue lengths;
                        if type 1 customer
                                if (all servers busy) add customer to queue
                                else{  update $D_1$;
                                        if server of type A is available {
                                                start service on type A server;
                                                schedule departure type 1 from type A
                                                server}
                                        else{  start service on type B server;
                                                schedule departure type 1 from type B
                                                server }
                        }
                    }
                else /* type 2 customer */
                    if (server A and B available){
                        update $D_2$;
                        start service;
                        schedule departure2;
                    }
                    else add customer to queue
            }

        departure1A:
            {
                update busy time of servers and total queue lengths;
                set server idle;
                if (server A and B are available and queue2 not empty)
                {
                    update $D_2$;
                    start service type 2 customer;
                    schedule departure2;
                }
                else
                    if queue1 is not empty
                    {
                        update $D_1$;
                        start service type 1 customer;
                        schedule new departure1A
                    }

            }

        departure 1B:
              similar

        departure2: /* almost the same */
            {
                update busy time of servers and total queue lengths,
                set one more server A and set server B idle
                if (queue2 not empty)
                {
                    update $D_2$;
                    start service type 2 customer;
                    schedule new departure2
                }
                else
                    if queue1 is not empty
                    {
                        update $D_1$;
                        start service type 1 customer on server A;
                        schedule departure1A;
                        if queue1 is still not empty
                        {
                            update $D_1$;
                            start service type 1 customer onserverB;
                            schedule departure1B
                        }
                    }

            }

    }
```

NB: To determine the amount of time that each of the servers spends on customers of type 1 and 2, we also need to know for each departure from a server A, from which of the two servers this departure takes place. This detail is not included in the pseudo-code.