Name: ................................................

Directions: Frequently save your work to handin and your USB key! No quizzes accepted with timestamp past 10:55 a.m. Submit a .tar file (even though just one file will be in it) with naming convention as in the homework (but with only names of those present).

1. (100)

Here you will modify the M/M/1 queue example included in the DES package.

The difference is simple. Instead of a First Come, First Served queuing policy, we will use a Shortest Time First policy: Whenever the server becomes free, it will choose from the queue that job that has the smallest service time. It is assumed that the service time is known upon arrival of the job to the system.

Note carefully: Your code does NOT have to be efficient. Just get code that works correctly. In particular, I suggest that you do NOT have your code maintain sorted order in the queue.
Solution:

library (DES)
# Shortest Time First scheduling; assumes that job service time is known
# upon arrival; this will be required to be element 5 in event objects

# assume appendfcfs() is still used to add to queue; this is probably
# faster in R, as delstf() can use which.min(), at C level

delstf <- function(queue) {
    # which is shortest?
    st <- which.min(queue$m[,5])
    nextevnt <- queue$m[st,]
    queue$m <- queue$m[-st,drop=FALSE]
    nextevnt
}

# STF version of M/M/1 queue example from package
mmlstf <- function(meaninterarrv,meansrv,timelim,dbg=FALSE) {

    # set up structures
    simlist <- newsim(dg)
    simlist$reactevent <- mmlreactstf
    simlist$arrvrate <- 1 / meaninterarrv
    simlist$srvrate <- 1 / meansrv
    simlist$srvbusy <- 0
    simlist$totwait <- 0.0
    simlist$queue <- newqueue(5)
    simlist$srvbusy <- FALSE
    simlist$jobnum <- 0
    simlist$arrvevnt <- 1
    simlist$srvevnt <- 2
    timetolstarriavl <- rexp(1, simlist$arrvrate)
    jobnum <- incremjobnum(simlist)

    # in this app, each event will be a 5–tuple, consisting of the basic
    # 2 (event time, type) plus arrival time, jobnum and service time
    schedevnt(timetolstarriavl, simlist$arrvevnt, simlist,
        c(timetolstarriavl, jobnum,NA))

    mainloop(simlist, timelim)

    cat("mean wait: ")
    print(simlist$totwait / simlist$totjobs)
}

incremjobnum <- function(simlist) {
    jobnum <- simlist$jobnum + 1
    simlist$jobnum <- jobnum
    jobnum
}

# what new events are triggered by the occurrence of an old one?
mmlreactstf <- function(evnt, simlist) {
    etype <- evnt[2]
    if (etype == simlist$arrvevnt) { # job arrival
        # determine service time upon arrival
        srvduration <- rexp(1, simlist$srvrate)
        evnt[5] <- srvduration
        # schedule next arrival
        timeofnextarrival <- simlist$currtime + rexp(1, simlist$arrvrate)
        jobnum <- incremjobnum(simlist)
        schedevnt(timeofnextarrival, simlist$arrvevnt, simlist,
            c(timeofnextarrival, jobnum,NA))
        # start newly–arrived job or queue it
        if (!simlist$srvbusy) { # server free, start job service
            simlist$srvbusy <- TRUE
            schedevnt(simlist$currtime + srvduration,
                simlist$srvvent,
                simlist, evnt[3:5]) # copy over previous data for this job
        }
    }

else {  # server busy, add job to queue
    appendfcfs(simlist$queue, evnt)
}
else if (etype == simlist$srvevt) {  # job completion
    # bookkeeping
    simlist$totjobs <- simlist$totjobs + 1
    # wait time = job completion time - job arrival time
    simlist$totwait <- simlist$totwait + simlist$currt ime - evnt[3]
    simlist$srvrbusy <- FALSE
    # check queue for waiting jobs
    if (nrow(simlist$queue$m) > 0) {  # nonempty queue
        qhead <- delstf(simlist$queue)
        # start job service
        simlist$srvrbusy <- TRUE
        srvduration <- qhead[5]
        schedevnt(simlist$currt ime + srvduration, simlist$srvevt, simlist,
                   qhead[3:4])
    }
}

