```
function possible = LEM_LSD(possible0,speed,lane_list,...
    lane_lengths,reservations,ht)
% LEM_LSD - lane strategic deconfliction flight reservation
% On input:
%     possible0 (1x2 vector): min and max possible start times
%     speed (float): desired speed in interval
%     lane_list (1xn vector): lane sequence
%     lane_lengths (1xn vector): length of lanes
%     reservations (reservations struct): flight reservation info
%     ht (float): headway time
% On output:
%     possible (interval set): possible launch intervals
% Call:
%     pos = LEM_LSD([0,30],5,[70,24,28,45],[10,10,10,10],reserv,20);
% Author:
%     T. Henderson
%     UU
%     Fall 2020
%

len_lane_list = length(lane_list);
intervals = possible0;
offset = 0;
c = 0;
total_time = 0;
dd = 0;
while ~isempty(intervals)&c<len_lane_list
    c = c + 1;
    lane_index = lane_list(c);
    dc = lane_lengths(lane_index);
    ts = dc/speed;
    [num_intervals,dummy] = size(intervals);
    for k = 1:num_intervals
        intervals(k,:) = intervals(k,:) + [offset,offset];
    end
    if dd>0
        intervals
    end
    c_flights = reservations(lane_index).flights;
    if ~isempty(c_flights)
        [num_c_flights,dummy] = size(c_flights);
```

```
        f = 0;
        [num_intervals,dummy] = size(intervals);
        while f<num_c_flights&~isempty(intervals)
            f = f + 1;
            tr1 = min(intervals(:,1));
            tr2 = max(intervals(:,2));
            ts1 = c_flights(f,2);
            ts2 = c_flights(f,3);
            tr1e = tr1 + ts;
            tr2e = tr2 + ts;
            if ~((ts1+ht<=tr1&ts2+ht<=tr1e)|(ts1-ht>=tr2&ts2-ht>=tr2e))
                new_intervals = [];
                for k = 1:num_intervals
                    k_intervals = LEM_OK_sched_req_enum(c_flights(f,↙
2),...
                        c_flights(f,3),c_flights(f,4),intervals(k,1),...
                        intervals(k,2),speed,dc,ht);
                    new_intervals = LEM_merge_intervals(k_intervals,...
                        new_intervals);
                end
                intervals = new_intervals;
                if isempty(intervals)
                    num_intervals = 0;
                else
                    num_intervals = length(intervals(:,1));
                end
            end
        end
        tch = 0;
    end
    offset = ts;
    total_time = total_time + ts;
end
total_time = total_time - ts;
[num_intervals,dummy] = size(intervals);
for k = 1:num_intervals
    intervals(k,:) = intervals(k,:) - [total_time,total_time];
end
if ~isempty(intervals)
    t1 = intervals(1,1);
    offset = 0;
```

```
    for c = 1:len_lane_list
        lane_index = lane_list(c);
        if ~isempty(reservations(lane_index).flights)...
                &abs(t1-reservations(lane_index).flights(1,1))<7
            tch = 0;
        end
        t1 = t1 + lane_lengths(c)/speed;
    end
end

possible = intervals;
```