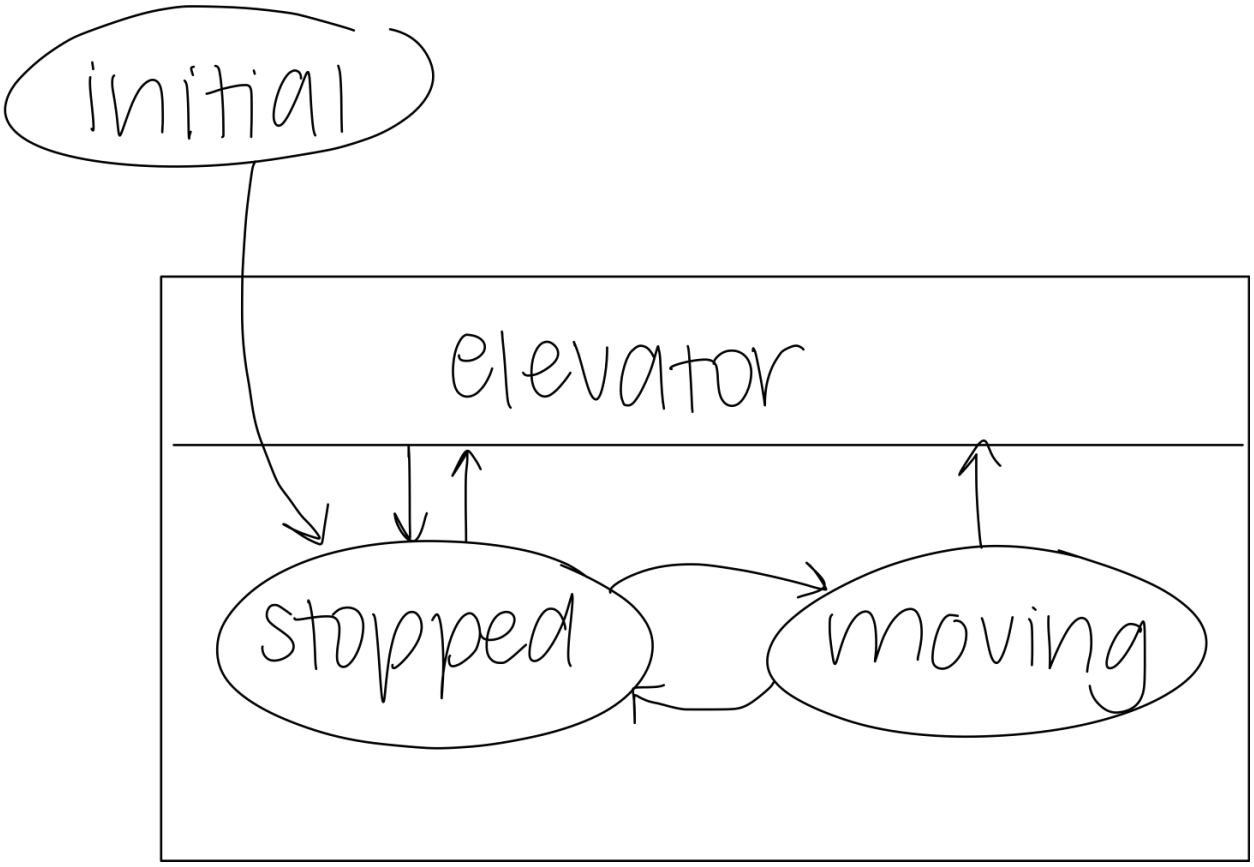
ECE 153A HW 4

Problem 1

Part A



Part B

The input signals for the FSM are:

1. Q_ENTRY_SIG
2. Q_INIT_SIG
3. Q_EXIT_SIG
4. TICK_SIG
5. F1_SIG
6. F2_SIG
7. F3_SIG
8. F4_SIG
9. F5_SIG
10. TERMINATE_SIG
11. PRINT_SIG

The state variables of the QP FSM are:

1. int floor_req_curr[5]: instantaneous request list
2. int floor_pen[5]: floors to stop at list
3. int curr_dir: instantaneous direction of travel
4. int curr_floor: location (updates on arrival event)
5. int stop_time: counts time to stop on a floor
6. int move_time: counts time between floors
7. int floor_curr_call_time[5]: measures when request is made
8. int floor_calls[5]: keeps track of how many requests were made to each floor
9. double floor_total_time[5]: keeps track of cumulative time to service each request
10. state: internal variable of QHsm that stores a pointer to the current state handler of the FSM

Part C

The explanations for the different QP Functions are as follows:

1. Q_TRAN(x): triggers a state transition and transitions from one elevator state to another (e.g. stopped → elevator) (where 'x' is a function pointer)
2. Q_SUPER(): Links back to superstate/parent state to handle the event, which allows shared behavior across substates - do this when current state can't handle signal
3. Q_HANDLED(): Declares an event handled and stays in the same state, awaits incoming signals so event is not reprocessed in parent state

Part D

When the elevator is stopped and receives a call for any floor, the FSM sets its state variables. It updates BSP_display, and if the floor is not already pending, it:

- Marks requested (floor_req_curr[floor_index]=1)
- Marks pending (update floor_pen & floor_req)
- Stores time request is made
- Increments num of calls to floor

It then returns Q_HANDLED().

Part E

When the elevator is stopped and receives a TICK_SIG, it updates BSP_display and checks if current floor is pending.

If current floor is pending:

- Wait at floor
- Increment time we've spent stopped till we hit max stopped time if not already there
- If at max stopped time check for other pending floors - if there are pending, transition to moving after clearing stop_time counter and pending status of current floor

The elevator switches to a moving state if there are other floors pending. Aka, if both of the below conditions are true:

- Current floor is not pending OR stop time at current floor is maxed out
- Another floor is pending

Part F

If TICK_SIG received while moving:

- BSP_display updated
- If move time not the total move time needed, increment move time counter
- Else
    - reset move time to 0 and
    - update curr_floor based on direction
- If curr floor pending switch to stopped state
- Keep moving and update pending so that calls to current floor only make the elevator stop when the calls are made when elevator is at least one floor away. When updating pending:
    - if elevator is moving upward
        - check for pending upward floors
        - if none, then check for pending downward floors
    - if elevator is moving downward
        - first check for pending downward floors
        - then check for pending upward floors

The elevator switches to the stopped state when the elevator reaches a pending floor.

Question 2

The average time between call and arrival for calls to each floor are as below:

Floor #	200 sec call time	100 sec call time	50 sec call time	20 sec call time	10 sec call time
1	9.65	10.01	10.02	15.12	43.65
2	7.07	6.99	6.92	11.14	27.59
3	6.09	5.94	6.10	10.16	21.42
4	6.97	7.02	7.03	11.15	27.58
5	10.04	10.00	10.01	15.12	43.58

Question 3

Simulated people entering by setting stop time to 5+(rand(1-10, 1 for each additional person up to 10 people)) → stop time = random number between 6 and 15

Re-averaging gave me:

Floor #	200 sec call time	100 sec call time	50 sec call time	20 sec call time	10 sec call time
1	10.06	10.23	10.05	16.72	47.48
2	7.26	6.92	6.91	12.74	30.40
3	6.02	6.28	6.09	11.29	24.23
4	7.14	7.09	6.98	12.57	30.57
5	9.78	10.02	9.96	16.50	47.89

Question 4

Setting the elevator simulation time to 500 seconds at 20 sec/call, with emergency key triggering at the end of this simulation time. The value we generate and average over and over is the time from this point (emergency trigger) till floor 1 is reached and the pending buttons are cleared. From rerunning this process 500 times, the average time to service an emergency call was **12.31 seconds.**

Question 5

Redundancies / implementation issues

- Different states are performing the same signal handling for floor calls - code is being reused in different floor calls
- This is not real hierarchical code as there is no proper use of superstates and substates

Fixes

- We can abstract more of the signal handling into handlers and have all the different states under one state
- We can simplify this into real hierarchical code by actually creating subsections of code that actively use their superstates as handlers to deal with events - essentially, we can move reused logic from substates with shared superstates into their superstate