Anderson, Christopher
Bronco ID: 014105585

Lab 3 Report

Part 1:

```
(:types passenger - object
        floor - object
        elevator - object
       )
```

```
(lift-at ?elevator - elevator ?floor - floor) 3👁 2☑ 2☒
;; current position of the lift is at ?floor
)
```

```
(:action stop
  :parameters (?e - elevator ?f - floor)
  :precondition (lift-at ?e ?f)
  :effect (and
            (forall (?p - passenger)
               (when (and (boarded ?e ?p)
                          (destin ?p ?f))
                     (and (not (boarded ?e ?p))
                          (served  ?p))))
            (forall (?p - passenger)
                (when (and (origin ?p ?f) (not (served ?p)))
                      (boarded ?e ?p)))))


;;drive up

(:action up
  :parameters (?e - elevator ?f1 - floor ?f2 - floor)
  :precondition (and (lift-at ?e ?f1) (above ?f1 ?f2))
  :effect (and (lift-at ?e ?f2) (not (lift-at ?e ?f1))))


;;drive down

(:action down
  :parameters (?e - elevator ?f1 - floor ?f2 - floor)
  :precondition (and (lift-at ?e ?f1) (above ?f2 ?f1))
  :effect (and (lift-at ?e ?f2) (not (lift-at ?e ?f1))))
)
```

Anderson, Christopher
Bronco ID: 014105585

1) To add support for multiple elevators I first added an elevator objects to the types section, this will allow for individual elevators to interact within the system. Next, Lift-at was modified to allow each elevator object to be set at a starting floor and track which floor each is currently on. After, all actions were modified by adding in the elevator object, this allowed for each elevator to take actions individually.

Problem for part 1:

```
(:objects p0 p1 p2 p3 p4 - passenger
          e0 e1 e2 - elevator
          f0 f1 f2 f3 f4 f5 f6 f7 - floor)
```

```
(origin p0 f7)
(destin p0 f6)

(origin p1 f6)
(destin p1 f0)

(origin p2 f5)
(destin p2 f0)

(origin p3 f4)
(destin p3 f0)

(origin p4 f3)
(destin p4 f0)

;;elevators
(lift-at e0 f0)
(lift-at e1 f0)
(lift-at e2 f0)
```

```
up e2 f0 f6
stop e2 f6
down e2 f6 f0
stop e2 f0
stop e0 f0
up e1 f0 f3
up e1 f3 f6
stop e1 f6
up e1 f6 f7
stop e1 f7
down e1 f7 f6
stop e1 f6
up e0 f0 f3
stop e2 f0
up e0 f3 f4
up e0 f4 f5
stop e0 f5
down e0 f5 f0
stop e0 f0
up e2 f0 f3
up e2 f3 f4
stop e2 f4
down e2 f4 f0
stop e2 f0
up e0 f0 f3
stop e0 f3
down e0 f3 f0
stop e0 f0
```

For this planner I have 5 people and 3 elevators, all elevators start at floor 0 for this test. First e2 goes to f6 and picks up p1 and drops them off at their destination f0. Next f1 goes up to f7 picks up p0 and drops them off at f6. After, e0 goes up to f5 picks up p2 and drops them off at f0. Next e2 goes to f4 picks up and drops off at f0 and e0 goes to f3 and drops off at f0.

Anderson, Christopher
Bronco ID: 014105585

Summary of planner:

All three elevators made were used in the planner, they seemed to alternate taking turns on which was used. Some strange behavior I noticed was the elevators not in use first would initialize using the stop action to what ever floor they started at. Also, they elevators would make multiple movement commands on floors that had people on them but decided not to stop and pick them up.

Part 2- Restrictions:

```
(can-travel ?elevator - elevator ?floor - floor) 2👁
    (boarded ?e - elevator ?p - passenger)
)
```

```
(:action servedpassenger
  :parameters (?e - elevator ?f - floor)
  :precondition (lift-at ?e ?f)
  :effect (and
            (forall (?p - passenger) (when (and (boarded ?e ?p) (destin ?p ?f)) (and (not (boarded ?e ?p)) (served  ?p)))))
)
;;Load
(:action board
  :parameters (?e - elevator ?f - floor)
  :precondition (lift-at ?e ?f)
  :effect (and
            (forall (?p - passenger) (when (and (origin ?p ?f) (not (served ?p))) (boarded ?e ?p))))
)
;;stop and drop
(:action dropoff
  :parameters (?e - elevator ?f - floor)
  :precondition (lift-at ?e ?f)
  :effect (and
          (forall (?p - passenger)
            (when
              (and (boarded ?e ?p) (not (served ?p)))
              (and (not (boarded ?e ?p)) (origin ?p ?f))

            )
          )
        )

)
;;drive up

(:action up
  :parameters (?e - elevator ?f1 - floor ?f2 - floor)
  :precondition (and (lift-at ?e ?f1) (above ?f1 ?f2) (can-travel ?e ?f2))
  :effect (and (lift-at ?e ?f2) (not (lift-at ?e ?f1))))


;;drive down

(:action down
  :parameters (?e - elevator ?f1 - floor ?f2 - floor)
  :precondition (and (lift-at ?e ?f1) (above ?f2 ?f1) (can-travel ?e ?f2))
  :effect (and (lift-at ?e ?f2) (not (lift-at ?e ?f1))))
)
```

For part2 I had to add a can-travel condition, basically all this was that it allowed for setting specific floors each elevator could travel to, allowing limitations of elevators. Similar to the floors this would have to be set for each elevator for each floor. I also added limitations on the up and down actions making sure it accounted for can-travel, as a precondition it would also check if the destination floor was

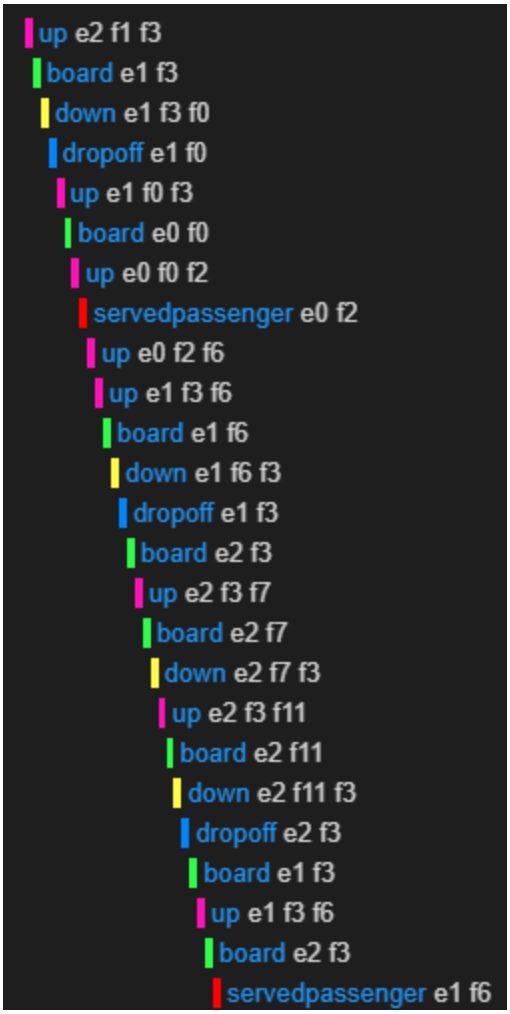
accessible by that elevator. The last things added was a dropoff action as well as a board action. The dropoff action is basically the same as stop but with a different effect. If a person was boarded and not yet served, they could then be, not boarded and have their origin change, in this case the floor that they were dropped off on. From here another elevator that had access to that floor could pick them up and continue their way to their destination. The final action I made was for planner readability since the output was much longer for this. I split stop into two different actions, servedpassanger and board, this way I could tell exactly what the planner was doing.

```
;;People

(origin p0 f7)
(destin p0 f0)

(origin p1 f6)
(destin p1 f1)

(origin p2 f5)
(destin p2 f3)

(origin p3 f4)
(destin p3 f7)

(origin p4 f3)
(destin p4 f2)

(origin p5 f11)
(destin p5 f6)

(origin p6 f10)
(destin p6 f9)
```

```
;;elevators

;;Even elevator
(lift-at e0 f0)
(can-travel e0 f0)
(can-travel e0 f2)
(can-travel e0 f4)
(can-travel e0 f6)
(can-travel e0 f8)
(can-travel e0 f10)

;;Divisible by 3 elevator
(lift-at e1 f3)
(can-travel e1 f0)
(can-travel e1 f3)
(can-travel e1 f6)
(can-travel e1 f9)

;;Odd Elevator
(lift-at e2 f1)
(can-travel e2 f1)
(can-travel e2 f3)
(can-travel e2 f5)
(can-travel e2 f7)
(can-travel e2 f9)
(can-travel e2 f11)
```

Listed above is how the implementation was done in the problem .ppdl, for this problem there is 7 people listed as p0-p6 and 3 elevators each with their limited can-travel floors manually entered. For divisible by 3 I used 0, 3, 6, 9 as my floors were labeled 0-11 for a total of 12 floors.

Anderson, Christopher
Bronco ID: 014105585



Here is a break down of travel pathing, if you look at the orgin/dest column you'll notice most of them are actually setup in a complicated manner where multiple elevators must be used.

Elevator start positions: e0(even) – f0, e1(div 3) – f3, e2(odd) – f1

Person	Orgin/dest	Floor moved to by/elevator						
P0	7/0	11,3/e2	0/e1					
P1	6/1	3/e1	7,11,3/e2	1/e2				
P2	5/3	3/e2						
P3	4/7	0/e0	9/e1	1,3,7/e2				
P4	3/2	0/e1	2/e0					
P5	11/6	3/e2	6/e1					
P6	10/9	0/e0	0/e1					

Above is a rough trace out of how each passenger reached their destination based off the planner. Cells marked in red text is when they got served and by which elevator. The elevators actually worked much better in this problem than the first, I'm not completely sure if this was due to limitations or splitting up actions but they seemed much more efficient allowing for multiple passengers in at once. All three

elevators worked together in setting each other up to allow each person to reach their destination, as seen in cells with black text you can view exactly how they traveled. No person had to use more than 3 rounds of swapping elevators- which is really good, they also never had to use the same one twice(without a different elevator in-between).