CS814 COURSEWORK 3

ANA HERNANDEZ

PART I

Question 1 – Domain

```
(define (domain cw3)
   (:requirements :typing)
   (:types key parcel car position)
   (:predicates
      (at-robby ?g - position)
      (at-keys ?k - key ?g - position)
      (at-parcel ?p - parcel ?g - position)
      (at-car ?c - car ?g - position)
      (keys-in-hand ?k - key)
      (free-keys ?k - key)
      (in-car ?c - car)
      (out-car ?c - car)
      (parcel-in-hand ?p - parcel)
      (free-parcel-hands ?p - parcel)
      (parcel-in-car ?p - parcel)
      (free-parcel-car ?p - parcel)
      (neighbour ?g1 ?g2 - position)
   )

   (:action walk
      :parameters (?fromG ?toG - position ?k - key)
      :precondition (and
         (neighbour ?fromG ?toG)
         (at-robby ?fromG)
         (free-keys ?k)
         )
      :effect (and
```

```
        (at-robby ?toG)
        (not(at-robby ?fromG)))
)

(:action grab_keys
    :parameters (?g - position ?k - key)
    :precondition (and
        (at-robby ?g)
        (at-keys ?k ?g)
        (free-keys ?k))
    :effect (and
        (keys-in-hand ?k)
        (not(free-keys ?k)))
)

(:action walk_with_keys
    :parameters (?fromG ?toG - position ?k - key ?c -car)
    :precondition (and
        (neighbour ?fromG ?toG)
        (at-robby ?fromG)
        (at-keys ?k ?fromG)
        (keys-in-hand ?k)
        (out-car ?c))
    :effect (and
        (at-robby ?toG)
        (not (at-robby ?fromG))
        (at-keys ?k ?toG)
        (not (at-keys ?k ?fromG)))
)

(:action drop_keys
    :parameters (?k - key ?g - position)
```

```
   :precondition (and
      (at-robby ?g)
      (at-keys ?k ?g)
      (keys-in-hand ?k))
   :effect (and
      (not(keys-in-hand?k))
      (free-keys ?k))
)

(:action get_in_car
   :parameters (?g - position ?k - key ?c - car)
   :precondition (and
      (at-robby ?g)
      (at-car ?c ?g)
      (at-keys ?k ?g)
      (out-car ?c)
      (keys-in-hand ?k))
   :effect (and
      (in-car ?c)
      (not(out-car ?c)))
)

(:action get_out_car
   :parameters (?c - car)
   :precondition
      (in-car ?c)
   :effect (and
      (not(in-car ?c))
      (out-car ?c))
)

(:action drive
```

```
    :parameters (?c - car ?fromG ?toG - position ?k - key ?p - parcel)
    :precondition (and
        (at-car ?c ?fromG)
        (at-robby ?fromG)
        (at-keys ?k ?fromG)
        (neighbour ?fromG ?toG)
        (in-car ?c)
        (keys-in-hand ?k)
        (free-parcel-car ?p)
        (free-parcel-hands ?p))
    :effect (and
        (at-robby ?toG)
        (not(at-robby ?fromG))
        (at-car ?c ?toG)
        (not(at-car ?c ?fromG))
        (at-keys ?k ?toG)
        (not(at-keys ?k ?fromG)))
)

(:action collect-parcel
    :parameters (?p - parcel ?c - car ?g - position)
    :precondition (and
        (at-robby ?g)
        (at-parcel ?p ?g)
        (at-car ?c ?g)
        (out-car ?c)
        (free-parcel-hands ?p)
        (free-parcel-car ?p))
    :effect (and
        (parcel-in-hand ?p)
        (not(free-parcel-hands ?p)))
)
```

```
(:action load-parcel-in-car
   :parameters (?p - parcel ?c - car ?g - position)
   :precondition (and
      (at-car ?c ?g)
      (at-robby ?g)
      (at-parcel ?p ?g)
      (out-car ?c)
      (parcel-in-hand ?p)
      (free-parcel-car ?p))
   :effect (and
      (parcel-in-car ?p)
      (not(parcel-in-hand ?p))
      (not(free-parcel-car ?p)))
)

(:action drive-with-parcel
   :parameters ( ?fromG ?toG - position ?p - parcel ?c - car ?k - key)
   :precondition (and
      (neighbour ?fromG ?toG)
      (in-car ?c)
      (at-car ?c ?fromG)
      (at-robby ?fromG)
      (at-keys ?k ?fromG)
      (keys-in-hand ?k)
      (at-parcel ?p ?fromG)
      (parcel-in-car ?p))
   :effect (and
      (at-robby ?toG)
      (not(at-robby ?fromG))
      (at-car ?c ?toG)
      (not(at-car ?c ?fromG))
```

```
            (at-keys ?k ?toG)
            (not(at-keys ?k ?fromG))
            (at-parcel ?p ?toG)
            (not(at-parcel ?p ?fromG)))
    )

    (:action load-parcel-out-car
        :parameters (?g - position ?p - parcel ?c - car)
        :precondition (and
            (at-robby ?g)
            (at-car ?c ?g)
            (at-parcel ?p ?g)
            (out-car ?c)
            (parcel-in-car ?p))
        :effect (and
            (parcel-in-hand ?p)
            (free-parcel-car ?p)
            (not(parcel-in-car ?p)))
    )

    (:action deliver-parcel
        :parameters (?p - parcel ?c - car ?g - position )
        :precondition
            (parcel-in-hand ?p)
        :effect (and
            (not (parcel-in-hand ?p))
            (free-parcel-hands ?p))
    )
)
```

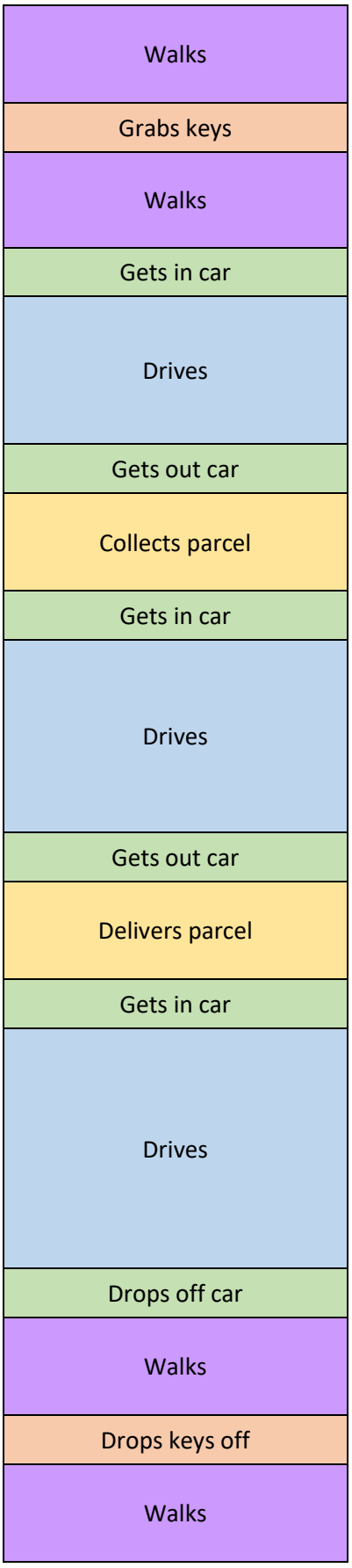
Question 2 – Problem

```
(define (problem cw3)
   (:domain cw3)
   (:objects key - key
      car - car
      parcel - parcel
      g1 g2 g3 g4 g5 g6 g7 g8 g9 g10 - position
   )
   (:init
      (at-robby g1)
      (at-keys key g4)
      (at-parcel parcel g6)
      (at-car car g3)
      (out-car car)
      (free-keys key)
      (free-parcel-hands parcel)
      (free-parcel-car parcel)

      (neighbour g1 g2)(neighbour g2 g3)(neighbour g2 g4)(neighbour g2 g5)
      (neighbour g5 g6)(neighbour g5 g7)(neighbour g7 g8)(neighbour g7 g9)
      (neighbour g9 g10)
      (neighbour g2 g1)(neighbour g3 g2)(neighbour g4 g2)(neighbour g5 g2)
      (neighbour g6 g5)(neighbour g7 g5)(neighbour g8 g7)(neighbour g9 g7)
      (neighbour g10 g9)
   )
   (:goal (and
      (at-robby g1)
      (at-keys key g4)
      (at-parcel parcel g10)
      (at-car car g3)
   ))
)
```

Question 3 – Plan

(walk g1 g2 key)
(walk g2 g4 key)
(grab_keys g4 key)
(walk_with_keys g4 g2 key car)
(walk_with_keys g2 g3 key car)
(get_in_car g3 key car)
(drive car g3 g2 key parcel)
(drive car g2 g5 key parcel)
(drive car g5 g6 key parcel)
(get_out_car car)
(collect-parcel parcel car g6)
(load-parcel-in-car parcel car g6)
(get_in_car g6 key car)
(drive-with-parcel g6 g5 parcel car key)
(drive-with-parcel g5 g7 parcel car key)
(drive-with-parcel g7 g9 parcel car key)
(drive-with-parcel g9 g10 parcel car key)
(get_out_car car)
(load-parcel-out-car g10 parcel car)
(deliver-parcel parcel car g10)
(get_in_car g10 key car)
(drive car g10 g9 key parcel)
(drive car g9 g7 key parcel)
(drive car g7 g5 key parcel)
(drive car g5 g2 key parcel)
(drive car g2 g3 key parcel)
(get_out_car car)
(walk_with_keys g3 g2 key car)
(walk_with_keys g2 g4 key car)
(drop_keys key g4)
(walk g4 g2 key)
(walk g2 g1 key)



Question 3 – Plan

The plan of action isn't too intuitive to read as this displays the parameters needed for each action which can confuse the reader. For instance, walking action is displayed "walk g1 g2 key", when in reality, you don't need or move a key in this action, but to differentiate between "walking" and "walking with keys", keys had to be a parameter, to make sure that "free-keys" was met before "walking".

Apart from this, the solution is the most efficient and quick way of solving this problem and this plan was constructed within seconds. Some issues were encountered when debugging the domain, like realising that "neighbour" positions had to be defined both ways, for example: neighbour g1 g2, neighbour g2 g1. This issue took me a long while to fix as I wasn't aware it wouldn't have a dual effect (for both cells, regardless of the order).

Also, due to the specifications of the coursework for the robot to be outside the car when loading the parcel in and out the car, two functions were created to transition the parcel between: its initial location, the robot's hands, and the car (and vice versa for unloading). This is the reason why the plan explains "collecting parcel" (robot is out the car, and parcel is in hands) and "loading parcel" (parcel is not in hands, parcel is now in car).

Links

Instead of copy pasting the code into the PDDL editor, this link could be used to access my PDDL Editor file:

http://editor.planning.domains/#edit_session=fX7Fkjdfioa6CHL