```
Question 3.1
Fixed relations:
    floor(i,j).
    adjacent(i,j,k,l).

Dynamic relations:
    boulder(i,j).
    player(i,j).

Question 3.2
Preconditions:
    move(i,j,k,l):
        floor(k,l), ~boulder(k,l), adjacent(i,j,k,l).

    push(i,j,k,l,m,n):
        floor(m,n), ~boulder(m,n), boulder(k,l), adjacent(i,j,k,l),
        adjacent(k,l,m,n), i - k = k - m, j - l = l - n.

Effects:
    move(i,j,k,l):
        player(k,l), ~player(i,j).
    push(i,j,k,l,m,n):
        player(k,l), ~player(i,j), boulder(m,n), ~boulder(k,l).

Question 3.3:
Initial State:
    floor(1,6),floor(1,7),floor(1,8),floor(1,9),
    floor(2,2),floor(2,3),floor(2,4),floor(2,6),
    floor(2,7),floor(2,8),floor(2,9),floor(3,2),
    floor(3,3),floor(3,4),floor(3,6),floor(4,2),
    floor(4,3),floor(4,4),floor(4,6),floor(4,8),
    floor(5,4),floor(5,8),floor(6,1),floor(6,4),
    floor(6,5),floor(6,6),floor(6,7),floor(6,8),
    floor(7,1),floor(7,3),floor(7,4),floor(7,5),
    floor(7,7),floor(7,8),floor(8,1),floor(8,3),
    floor(8,4),floor(8,5),floor(9,1),floor(9,7),
    floor(9,8),floor(9,9),

    boulder(3,3),boulder(3,4),boulder(4,3),

    player(2,2),

    adjacent(2,2,2,3),adjacent(2,2,3,2)
    ''' #for each corner tile

    adjacent(2,3,2,2),adjacent(2,3,2,4),adjacent(2,3,3,3),
    ''' #for each wall adjacent tile that is not a corner

    adjacent(3,3,2,3),adjacent(3,3,3,2),adjacent(3,3,3,4),
     adjacent(3,3,4,3)
    ''' #for each not wall adjacent tile
```

Question 3.4:
Goal:
    boulder(4,8),boulder(5,8),boulder(6,8)

Question 3.5:
    The search space can be very large. Without specifying a search algorithm with a distance heustistic (like A*), an ASP solution can be slower than one where the search algorithm is defined explicitly.