Artificial Intelligence

Task 1

4a). **property_query(tweety,movement,fly).**

For this query, once the actual query has been called, hasprop is called and fails because hasprop(tweety,movement,fly) does not exist In file. It then backtracks to decide if 'tweety' has other properties, so isa(tweety,canary) is called to see if 'tweety' is a 'canary', which returns true because they have the same properties. It then exits and calls hasprop(canary,movement,fly) which fails because it does not exist in the file either. It then backtracks and calls isa(canary,bird) to see if a 'canary' is a 'bird', which returns true because they have the same properties. It exits and then calls hasprop(bird,movement,fly), which returns true because it does exist in the file. This means that 'canary' and 'tweety' have the fly and movement properties.

4b). **property_query(tweety,movement,walk).**

For this query, once the actual query has been called, hasprop(tweety,movement,walk) is called and fails because it does not exist in the file. It then backtracks and calls isa(tweety, canary) which returns true as they have the same properties. It then calls hasprop(canary, movement, walk) which returns false because it does not exist in the file. It then backtracks and calls isa(canary,bird) which returns true, as it does exist in the file and implies they have the same properties. It then calls hasprop(bird, movement, walk) which returns false as it does not exist in the file. In then calls isa(bird,anaimal) which return true, as this exists in the file. It then calls hasprop(animal, movement, walk) which fails as it does not exist. It then calls isa(animal,X), which returns as animal is the highest in the hierarchy, so then all of the other queries return false.

4c). **property_query(ostrich,movement,X).**

For this query, once the actual query has been called, hasprop(ostrich, movement, X) is called which finds the first property 'walk', then calls hasprop(ostrich, movement, walk) which returns true as it exists in the file. It then calls isa(ostrich, bird) which returns true because they have similar properties. It then calls hasprop(bird, movement,X) which finds the second property 'fly', then calls hasprop(bird, movement,fly) which returns true. It then backtracks to see if X can equal any other properties. It then calls hasprop(bird, movement, X), which fails as it doesn't exist in the file. It then backtracks and calls isa(bird, X) which finds the property 'animal' which returns true as they have similar properties. It then calls hasprop(animal, movement, X) which fails because it doesn't exist in the file. It then calls isa(animal, X) which fails as it doesn't exits. It then backtracks to see if X can equal anything else which fails therefore causing the query to fail as X can only be 'fly' and 'walk'.

4d). **property_query(animal,colour,X).**

For this query, once the actual query has been called, hasprop(animal, colour, X) which fails because colour doesn't share a relationship with animal.. It then calls isa(animal,X) which fails as there is nothing higher than 'animal' in the hierarchy, so therefore the rest of the query fails.

4d). **property_query(A,cover,feathers).**

Firstly for this query, once the actual query is called, hasprop(A, cover, feathers) which finds the first property 'bird', then it calls hasprop(bird, cover, feathers) which returns true. It then backtracks and calls isa(X, Y)which finds the properties 'tweety' and 'canary' respectively. It then calls isa(tweety, canary) which returns true as they have similar properties. It then calls hasprop(canary, cover, feathers) which fails, so it backtracks and calls isa(canary, X) which finds the first property 'bird' so it then calls isa(canary, bird) which returns because they have similar properties. It then calls hasprop(bird, cover, feathers) which returns true, so the query returns true. Secondly, isa(bird, X) is called which finds the first property 'animal', so isa(bird, animal) is called and returns true as they have similar properties. Then hasprop(animal, cover, feathers) is called and fails as it doesn't exist in the file. Prolog then backtracks and repeats this process with 'tweety', 'canary' and 'ostrich' as it has found bird, so it implies the rest are covered in feathers.

Task 2: Warehouse Management

1)

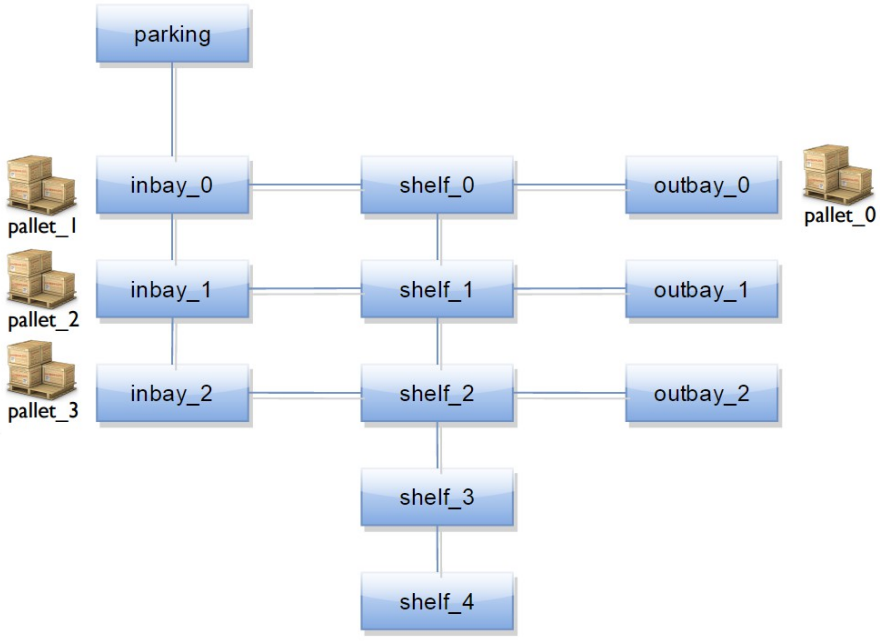
I will firstly state my types before the predicates are identified. The types are: place – object, movable – object, shelf – place, inbay – place, parking – place, outbay – place, forklift – movable and pallet – movable.

The predicates are:

- (connected ?a - place ?b - place)
  - *This defines that places in the warehouse are connected*
- (holding ?c - pallet)
  - *This defines if the forklift is currently holding a pallet*
- (at ?pallet - movable ?room - place)
  - *This defines if movable objects are stored at places in the warehouse*
- (clear ?empty - place)
  - *This defines if a place in the warehouse is clear of a movable object (pallet)*
- (available ?d - forklift)
  - *This defines if the forklift is available to pick something up*

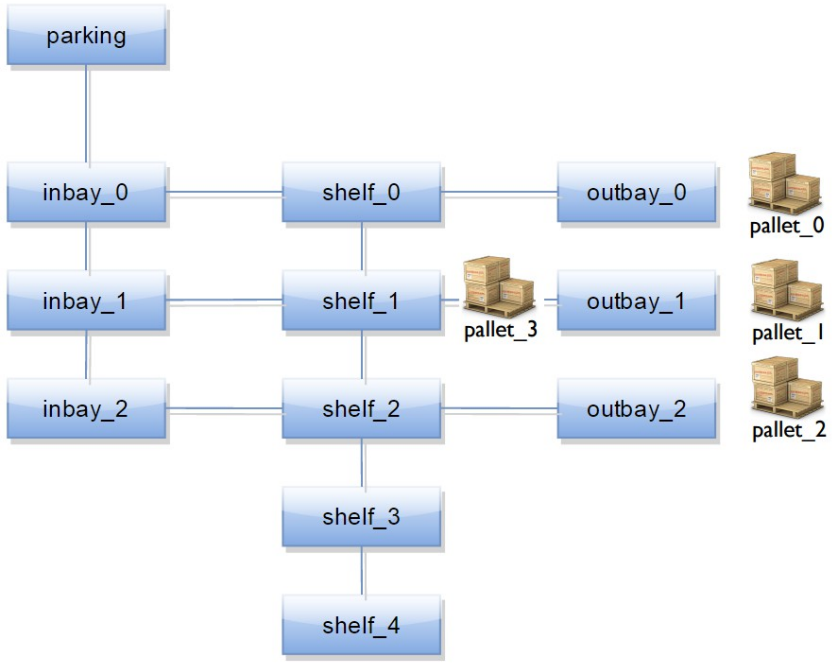
4)

Depot-01
Here is a diagram of my visualisation for depot-01.pddl. As the diagram demonstrates, the goal state is that 'pallet_0' should be at 'outbay_0'.



Depot-02

Here is a diagram of my visualisation for depot-02.pddl. As the diagram demonstrates, the goal state is that 'pallet_0' should be at 'outbay_0', 'pallet_1' should be in 'outbay_1', 'pallet_2 should be in 'outbay_2' and 'forklift_0' to be in 'parking'.



5)

Depot-01

For this problem to be solved, the domain first utilises the 'drive' action to move the forklift from 'parking' to 'inbay_0' and then again to move the forklift from 'inbay_0' to 'shelf_0'. It then utilises the 'pickup' action to pick up 'pallet_0' from 'shelf_0' into the forklift. It then utilises the 'drive' action again to move 'forklift_0' from 'shelf_0' to 'outbay_0'. Finally it utilises the 'setdown' action to place 'pallet_0' in 'outbay_0'. The total cost is 5.

Depot-02

For this problem to be solved, the domain first utilises the 'drive' action to move 'forklift_0' from 'parking' into 'inbay_0', then utilises the 'pickup' action to pick up 'pallet_1' from 'inbay_0'. It then utilises the 'drive' action to move 'forklift_0' from 'inbay_0' to 'shelf_0', then again to move from 'shelf_0' to 'shelf_1' and again to move from 'shelf_1' to 'outbay_1'. It then utilises the 'setdown' action to put 'pallet_1' in 'outbay_1'. It then utilises the 'drive' action to move from 'outbay_1' to 'shelf_1', then again from 'shelf_1' to 'inbay_1'. It then utilises the 'pickup' action to pick up 'pallet_2', then utilises the 'drive' action to drive to 'shelf_1', then again to 'shelf_2', and then finally to 'outbay_2'. It then utilises the 'setdown' action to put down 'pallet_2' in 'outbay_2'. It then utilises the 'drive' action to move from 'outbay_2' to 'shelf_2', then to 'inbay_2'. It then utilises the 'pickup' action to pick up 'pallet_3' , then utilises the 'drive' action' to move to 'shelf_2', then to 'shelf_1'. It then utilises the 'setdown' action to put 'pallet_3' in 'shelf_1', then utilises the 'drive' action to move to 'shelf_1' and then 'shelf_0'. It then utilises the 'pickup' to pick up 'pallet_0', then it utilises the 'drive' action to move from 'shelf_0' to 'outbay_0' It then utilises the 'setdown' action to put down 'pallet_0' in 'outbay_0'. It then utilises the 'drive' action to move from 'outbay_0' to 'shelf_0', then to 'inbay_0' and then finally to 'parking. The total cost is 26.