1.

a)

Environment: Partly observable, Non-deterministic, Dynamic, Discrete, Multiple agents.
Agent: Utility agent.

b)

Environment: Partly observable, Deterministic, Dynamic, Continuous, Single agent.
Agent: Utility agent.

c)

Environment: Partly observable, Deterministic, Dynamic, Continuous, Single agent.
Agent: Utility agent.

d)

Environment: Observable, Deterministic, Static, Discrete, Single agent.
Agent: Table lookup.

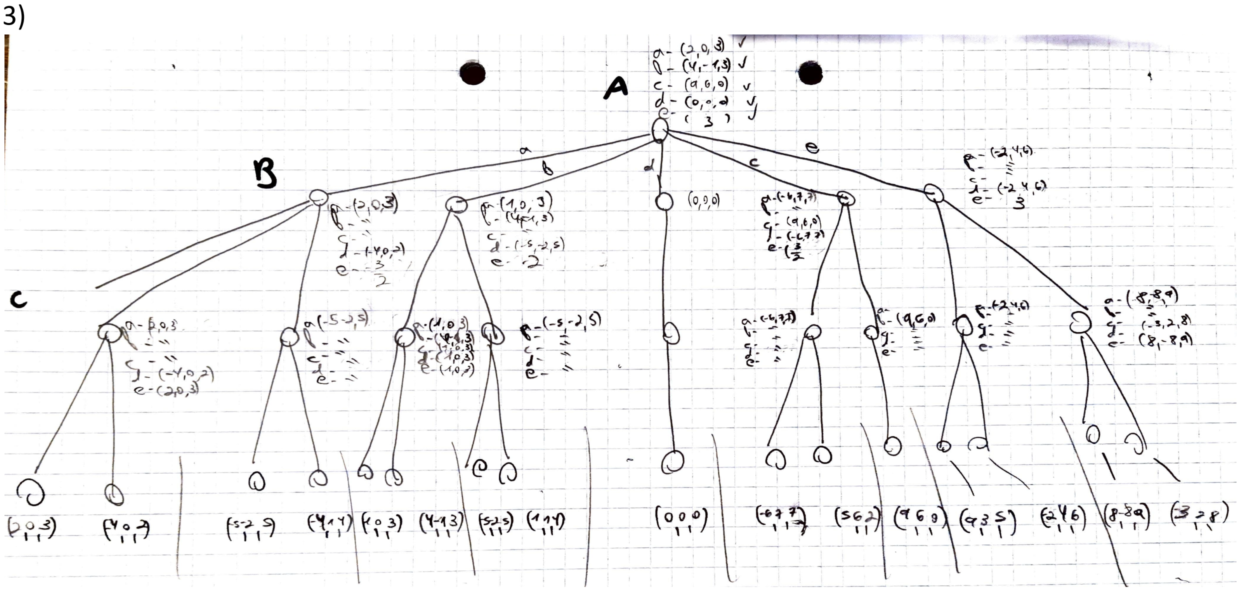
e)

Environment: Observable, Deterministic, Static, Discrete, Single agent.
Agent: Goal based.

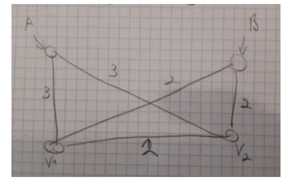
2)

a) Greedy - v0, v3, v2, v1
b) A* - v0, v2, v1, v3
c) RTA - v0, v2, v1, v3
d) The paths remain the same.
The heuristic is not admissible.
For example, in the last step of A*, the state is s= {currentVertex=v1, visited=[v0,v2]}, so the h'(s) = mst({v1,v3}) * 2 = 4. When the real path to goal is 2, so h'(s) > realPath.

3)



4) a) In the following graph, assuming that v1 and v2 have people on it, if A will play first, than B can see where he goes, and go to the same place and get there before him, and than A will rescue no people.
If he will do no-op, he could see where B went, and go the other vertex, and save at least part of the people.
b,c in aner file – “4-b,c”



5)

a.
Satisfiable – with model = {A,B,C,D, E,F =True}
Not valid – with model = {A=False, B,C,D,E,F=True} the sentence is False.
Number of models – we can notice that its not possible that both sides of the sentence’s or will be True together, because one contain A and one not A.
Models which the first half is true – {A,B,C,D,E=False, F=True} => 1 model.
Models which the second half is true – {A,B,C,D=True, E,F=ϕ} => 4 models.
Total – 5 models.

b. Satisfiable – with model = {A,B,C,D = True, E=False}
Not Valid – with model = { A,B,C,D,E = True } the sentence is False.
Number of models – We can see that there are $2^5 = 32$ models for this sentence.
For the sentence to fail we need all the variables to be the same, there are only 2 options for that, all True or all False.
So number of models = 32-2 = 30.

c. Unsatisfiable – if A=True that that last part of the and is False, and same if A=False.
Not Valid – because Unsatisfiable.
Number of models – 0.

d. Satisfiable – with model = {A,B,C,D = True }
Valid – If the left side of the => is True, meaning A,B,C,D=True, and than the right side will be also True, because D=True.
Otherwise, the sentence is True to all Models.
Number of models – $2^4 = 16$.

e. Unsatisfiable – with every model we will have the same result as section d, either the left side of the => is True, which makes C=True, so the left side of the => is True,
which make the whole sentence to be False (because of the not), or the left side of the => is False, and then the sentence always exist, and with the not its False.
Number of models – 0.

f. Satisfiable – with model = {A=False, B=True}
Not Valid – with model = {A=True, B=False }the sentence is False.
Number of models – The only model who satisfies is the one above,
So number of models = 1

6)

a.

1. Vertex(v)∧Loc(v,s) ∧ PeopleAt(v,s) ∧¬Terminated(s) => ¬PeopleAt(v,Result(Pick,s))
2. Loc(v,s) ∧ Edge(e,v,v’) ∧ Time(t,s) ∧Weight(e,w) ∧TimeLeft(t,e,s) => Loc(v’,Result(traverse(e),s)) ∧ Time(-(t,w), Result(traverse(e),s))
3. ¬PickAll(s) => ¬Terminated(s)
4. ∀v (Vertex(v)∧ ¬PeopleAt(v,s)) => PickAll(s)
5. Time(t,s) ∧ Deadline(t) => Terminated(Result(Terminate,s))
6. Weight(e,w) ∧ Time(t,s) ∧ Deadline(t’) ∧ >=(-(t’,t),e) => TimeLeft(t,e,s)

b.

a)

1. (¬Vertex(v)∨ ¬Loc(v,s) ∨¬PeopleAt(v,s) ∨Terminated(s)) ∨ ¬PeopleAt(v,Result(a,s))

2. (¬Loc(v,s) ∨ ¬Edge(e,v,v’) ∨ ¬Time(t,s) ∨¬Weight(e,w) ∨¬TimeLeft(t,e,s)) ∨
Loc(v’,Result(traverse(e),s)) ∧ Time(-(t,w), Result(traverse(e),s))

3. PickAll(s) ∨ ¬Terminated(s)

4. (¬ Vertex(V0) ∨ PeopleAt(V0,s) ∨ ¬Vertex(V1) ∨ PeopleAt(V1,s) ∨
¬Vertex(V2) ∨ PeopleAt(V2,s) ∨ ¬Vertex(V3) ∨ PeopleAt(V3,s))
∨ PickAll(s)

5. ¬Time(t,s) ∨¬ Deadline(t) ∨ Terminated(Result(Terminate,s))

6. ¬Weight(e,w) ∨ ¬Time(t,s) ∨ ¬Deadline(t’) ∨ ¬>=(-(t’,t),e) ∨ TimeLeft(t,e,s)

b)

Goal: PickAll(s), so negate is ¬PickAll(s)

We will remove all time connected axiomas, so new KB will look like -

1. ¬Vertex(v)∨ ¬Loc(v,s) ∨¬PeopleAt(v,s) ∨Terminated(s) ∨ ¬PeopleAt(v,Result(a,s))

2. ¬Loc(v,s) ∨ ¬Edge(e,v,v’) ∨¬Weight(e,w) ∨
Loc(v’,Result(traverse(e),s))

3. PickAll(s) ∨ ¬Terminated(s)

4. ¬ Vertex(V0) ∨ PeopleAt(V0,s) ∨ ¬Vertex(V1) ∨ PeopleAt(V1,s) ∨
¬Vertex(V2) ∨ PeopleAt(V2,s) ∨ ¬Vertex(V3) ∨ PeopleAt(V3,s)
∨ PickAll(s)

and by adding the givens –

Vertex(V0)

Vertex(V1)

Vertex(V2)

Vertex(V3)

Edge(E1, V0, V1)

Edge(E2, V0, V2)

Edge(E3, V2, V3)
Edge(E4, V1, V3)
Edge(E5, V1, V2)
Weight(E1,4)
Weight(E2,1)
Weight(E3,5)
Weight(E4,2)
Weight(E5,1)
Loc(V0, S0)
PeopleAt(V1, S0)
PeopleAt(V2, S0)
PeopleAt(V3, S0)
Proof in another file – 6-b_c.

c. We can see in the proof that it was not possible to prove without using "Frame axiomas", because there is a need to know the previous picked people vertices, and the route the agent does, and it is not possible to reconstruct the possibilities without holding states.
d. If we would want to use forward chaining, we would have to switch the KB to "Horn form" sentences. We have sentences that have negation, so we can't translate our KB to Horn form, meaning that we cannot use forward chaining.