1)

a) An agent that plays Poker.

Needs a utility-based agent to optimize the score.

This environment can be seen as deterministic (despite the randomness of dealing cards initially, as this randomness is "swallowed" by the inaccessibility), inaccessible, non-episodic (can learn opponents tactics, etc.), discrete, semi-static (not allowed to think forever, but nothing changes between moves), and multi-agent.

b) An agent that can play Tic-Tac-Toe.

ANSWER: Can be done by a goal-based agent, but since this is a very small domain can be "Compiled down" to a reflex agent, even using a table lookup.   
Environment is: deterministic, accessible, episodic, static, discrete, and multi-agent.

c) Robot to Win DARPA Challenge.

Utility-based.  
Environment is: stochastic, inaccessible, dynamic, non-episodic, and continuous. It is also made multi-agent due to the fact that there is an operator.

d) An internet coordination agent (sets meetings between humans with conflicting goals).

Utility based.   
The environment here is deterministic, inaccessible, (perhaps) non-episodic (issues of trust/dependability of a participant), dynamic, (practically) discrete (you CAN set a meeting at 3 and 5.1234 minutes, but that is almost never done!), and multi-agent.

e) An agent that can play peg-and-hole solitaire.

Goal based or utility based. Possibly can be compiled into rule-based and thus reflex, but domain is too large for straightforward table. Environment is: deterministic, accessible, static, episodic, and single-agent.

f) An agent that plays Go optimally.

Goal (search)-based.   
Environment is deterministic, accessible, discrete, episodic, static, and multi-agent.

2) vertex here are numbered from 1 to 3 instead of from 0 to 2.  
a)   
Pulled Node:Node@a418fc- vertex-1.0 supplies:1.0 newDepth:0.0  
Expanding Node:Node@a418fc- v1.0 supplies:1.0 Depth:0.0  
Create Node:Node@105068a- vertex-2.0 supplies:3.0 newDepth:1.0  
Create Node:Node@132e575- vertex-3.0 supplies:-1.0 newDepth:Infinity  
Create NoOp Node:Node@19f2327- vertex-1.0 supplies:0.0 newDepth:1.0

Pulled Node:Node@132e575- vertex-3.0 supplies:-1.0 newDepth:Infinity

Found solution Node: vertex-3.0 supplies:-1.0 newDepth:Infinity

Path of GreedySearchYazidi@1f4acd0:V3.0

b)  
Pulled Node:Node@105068a- vertex-1.0 supplies:1.0 newDepth:0.0  
Expanding Node:Node@105068a- v1.0 supplies:1.0 Depth:0.0  
Create Node:Node@132e575- vertex-2.0 supplies:3.0 newDepth:1.0

Create Node:Node@19f2327- vertex-3.0 supplies:-1.0 newDepth:Infinity

Create NoOp Node:Node@1f4acd0- vertex-1.0 supplies:0.0 newDepth:1.0

Pulled Node:Node@1f4acd0- vertex-1.0 supplies:0.0 newDepth:1.0

Expanding Node:Node@1f4acd0- v1.0 supplies:0.0 Depth:1.0

Create Node:Node@bedef2- vertex-2.0 supplies:2.0 newDepth:Infinity

Create Node:Node@af905d- vertex-3.0 supplies:-2.0 newDepth:Infinity

Create NoOp Node:Node@7716f4- vertex-1.0 supplies:-1.0 newDepth:Infinity

Pulled Node:Node@132e575- vertex-2.0 supplies:3.0 newDepth:1.0

Expanding Node:Node@132e575- v2.0 supplies:3.0 Depth:1.0

Create Node:Node@efb846- vertex-1.0 supplies:2.0 newDepth:4.0

Create Node:Node@84aee7- vertex-3.0 supplies:-1.0 newDepth:Infinity

Create NoOp Node:Node@9ee92- vertex-2.0 supplies:2.0 newDepth:2.0

Pulled Node:Node@9ee92- vertex-2.0 supplies:2.0 newDepth:2.0

Expanding Node:Node@9ee92- v2.0 supplies:2.0 Depth:2.0

Create Node:Node@f39991- vertex-1.0 supplies:1.0 newDepth:4.0

Create Node:Node@12b3a41- vertex-3.0 supplies:-2.0 newDepth:Infinity

Create NoOp Node:Node@574795- vertex-2.0 supplies:1.0 newDepth:3.0

Pulled Node:Node@efb846- vertex-1.0 supplies:2.0 newDepth:4.0

Expanding Node:Node@efb846- v1.0 supplies:2.0 Depth:4.0

Create Node:Node@16f4d33- vertex-2.0 supplies:4.0 newDepth:6.0

Create Node:Node@1fc625e- vertex-3.0 supplies:0.0 newDepth:8.0

Create NoOp Node:Node@199bd52- vertex-1.0 supplies:1.0 newDepth:5.0

Pulled Node:Node@f39991- vertex-1.0 supplies:1.0 newDepth:4.0

Expanding Node:Node@f39991- v1.0 supplies:1.0 Depth:4.0

Create Node:Node@1023edb- vertex-2.0 supplies:3.0 newDepth:5.0

Create Node:Node@71f4dd- vertex-3.0 supplies:-1.0 newDepth:Infinity

Create NoOp Node:Node@df9f5f- vertex-1.0 supplies:0.0 newDepth:5.0

Pulled Node:Node@574795- vertex-2.0 supplies:1.0 newDepth:3.0

Expanding Node:Node@574795- v2.0 supplies:1.0 Depth:3.0

Create Node:Node@5ede7b- vertex-1.0 supplies:0.0 newDepth:4.0

Create Node:Node@74cd4d- vertex-3.0 supplies:-3.0 newDepth:Infinity

Create NoOp Node:Node@1961c42- vertex-2.0 supplies:0.0 newDepth:4.0

Pulled Node:Node@5ede7b- vertex-1.0 supplies:0.0 newDepth:4.0

Expanding Node:Node@5ede7b- v1.0 supplies:0.0 Depth:4.0

Create Node:Node@1b54208- vertex-2.0 supplies:2.0 newDepth:Infinity

Create Node:Node@1e6f5c3- vertex-3.0 supplies:-2.0 newDepth:Infinity

Create NoOp Node:Node@1884d57- vertex-1.0 supplies:-1.0 newDepth:Infinity

Pulled Node:Node@1fc625e- vertex-3.0 supplies:0.0 newDepth:8.0

Found solution Node: vertex-3.0 supplies:0.0 newDepth:8.0

Path of AStarYazidi@a418fc:V2.0, V1.0, V3.0

c)  
Pulled Node:Node@105068a- vertex-1.0 supplies:1.0 newDepth:0.0  
Expanding Node:Node@105068a- v1.0 supplies:1.0 Depth:0.0  
Create Node:Node@132e575- vertex-2.0 supplies:3.0 newDepth:1.0  
Create Node:Node@19f2327- vertex-3.0 supplies:-1.0 newDepth:Infinity  
Create NoOp Node:Node@1f4acd0- vertex-1.0 supplies:0.0 newDepth:1.0

Pulled Node:Node@1f4acd0- vertex-1.0 supplies:0.0 newDepth:1.0

Expanding Node:Node@1f4acd0- v1.0 supplies:0.0 Depth:1.0

Create Node:Node@bedef2- vertex-2.0 supplies:2.0 newDepth:Infinity

Create Node:Node@af905d- vertex-3.0 supplies:-2.0 newDepth:Infinity

Create NoOp Node:Node@7716f4- vertex-1.0 supplies:-1.0 newDepth:Infinity

Path of RTAStarYazidi@a418fc:V1.0

RTAStarYazidi@a418fc Chose NoOp

Pulled Node:Node@efb846- vertex-1.0 supplies:0.0 newDepth:0.0

Expanding Node:Node@efb846- v1.0 supplies:0.0 Depth:0.0

Create Node:Node@84aee7- vertex-2.0 supplies:2.0 newDepth:Infinity

Create Node:Node@9ee92- vertex-3.0 supplies:-2.0 newDepth:Infinity

Create NoOp Node:Node@f39991- vertex-1.0 supplies:-1.0 newDepth:Infinity

Pulled Node:Node@9ee92- vertex-3.0 supplies:-2.0 newDepth:Infinity

Pulled Node:Node@84aee7- vertex-2.0 supplies:2.0 newDepth:Infinity

Pulled Node:Node@f39991- vertex-1.0 supplies:-1.0 newDepth:Infinity

No Solution Found.

d) Repeat a-c but using h'(n) = 2\*h(n) as the heuristic. Is h'(n) admissible?

ANSWER: h'(n) is not admissible. For example, it overestimates even for S0, for which h'(S0) = 8 while the cost of the optimal solution is 6.

Now this makes no difference for **case (a)** (greedy search) since multiplying all values by a positive constant does not change the order of expanded nodes. So no change in the search in case (a).

**case (b)**

Pulled Node:Node@105068a- vertex-1.0 supplies:1.0 newDepth:0.0

Expanding Node:Node@105068a- v1.0 supplies:1.0 Depth:0.0

Create Node:Node@132e575- vertex-2.0 supplies:3.0 newDepth:1.0

Create Node:Node@19f2327- vertex-3.0 supplies:-1.0 newDepth:Infinity

Create NoOp Node:Node@1f4acd0- vertex-1.0 supplies:0.0 newDepth:1.0

Pulled Node:Node@1f4acd0- vertex-1.0 supplies:0.0 newDepth:1.0

Expanding Node:Node@1f4acd0- v1.0 supplies:0.0 Depth:1.0

Create Node:Node@bedef2- vertex-2.0 supplies:2.0 newDepth:Infinity

Create Node:Node@af905d- vertex-3.0 supplies:-2.0 newDepth:Infinity

Create NoOp Node:Node@7716f4- vertex-1.0 supplies:-1.0 newDepth:Infinity

Pulled Node:Node@132e575- vertex-2.0 supplies:3.0 newDepth:1.0

Expanding Node:Node@132e575- v2.0 supplies:3.0 Depth:1.0

Create Node:Node@efb846- vertex-1.0 supplies:2.0 newDepth:4.0

Create Node:Node@84aee7- vertex-3.0 supplies:-1.0 newDepth:Infinity

Create NoOp Node:Node@9ee92- vertex-2.0 supplies:2.0 newDepth:2.0

Pulled Node:Node@9ee92- vertex-2.0 supplies:2.0 newDepth:2.0

Expanding Node:Node@9ee92- v2.0 supplies:2.0 Depth:2.0

Create Node:Node@f39991- vertex-1.0 supplies:1.0 newDepth:4.0

Create Node:Node@12b3a41- vertex-3.0 supplies:-2.0 newDepth:Infinity

Create NoOp Node:Node@574795- vertex-2.0 supplies:1.0 newDepth:3.0

Pulled Node:Node@574795- vertex-2.0 supplies:1.0 newDepth:3.0

Expanding Node:Node@574795- v2.0 supplies:1.0 Depth:3.0

Create Node:Node@16f4d33- vertex-1.0 supplies:0.0 newDepth:4.0

Create Node:Node@1fc625e- vertex-3.0 supplies:-3.0 newDepth:Infinity

Create NoOp Node:Node@199bd52- vertex-2.0 supplies:0.0 newDepth:4.0

Pulled Node:Node@16f4d33- vertex-1.0 supplies:0.0 newDepth:4.0

Expanding Node:Node@16f4d33- v1.0 supplies:0.0 Depth:4.0

Create Node:Node@1023edb- vertex-2.0 supplies:2.0 newDepth:Infinity

Create Node:Node@71f4dd- vertex-3.0 supplies:-2.0 newDepth:Infinity

Create NoOp Node:Node@df9f5f- vertex-1.0 supplies:-1.0 newDepth:Infinity

Pulled Node:Node@f39991- vertex-1.0 supplies:1.0 newDepth:4.0

Expanding Node:Node@f39991- v1.0 supplies:1.0 Depth:4.0

Create Node:Node@5ede7b- vertex-2.0 supplies:3.0 newDepth:5.0

Create Node:Node@74cd4d- vertex-3.0 supplies:-1.0 newDepth:Infinity

Create NoOp Node:Node@1961c42- vertex-1.0 supplies:0.0 newDepth:5.0

Pulled Node:Node@efb846- vertex-1.0 supplies:2.0 newDepth:4.0

Expanding Node:Node@efb846- v1.0 supplies:2.0 Depth:4.0

Create Node:Node@1b54208- vertex-2.0 supplies:4.0 newDepth:6.0

Create Node:Node@1e6f5c3- vertex-3.0 supplies:0.0 newDepth:8.0

Create NoOp Node:Node@1884d57- vertex-1.0 supplies:1.0 newDepth:5.0

Pulled Node:Node@1e6f5c3- vertex-3.0 supplies:0.0 newDepth:8.0

Found solution Node: vertex-3.0 supplies:0.0 newDepth:8.0

Path of AStarYazidi@a418fc:V2.0, V1.0, V3.0

**Case (c):**

Pulled Node:Node@105068a- vertex-1.0 supplies:1.0 newDepth:0.0

Expanding Node:Node@105068a- v1.0 supplies:1.0 Depth:0.0

Create Node:Node@132e575- vertex-2.0 supplies:3.0 newDepth:1.0

Create Node:Node@19f2327- vertex-3.0 supplies:-1.0 newDepth:Infinity

Create NoOp Node:Node@1f4acd0- vertex-1.0 supplies:0.0 newDepth:1.0

Pulled Node:Node@1f4acd0- vertex-1.0 supplies:0.0 newDepth:1.0

Expanding Node:Node@1f4acd0- v1.0 supplies:0.0 Depth:1.0

Create Node:Node@bedef2- vertex-2.0 supplies:2.0 newDepth:Infinity

Create Node:Node@af905d- vertex-3.0 supplies:-2.0 newDepth:Infinity

Create NoOp Node:Node@7716f4- vertex-1.0 supplies:-1.0 newDepth:Infinity

Path of RTAStarYazidi@a418fc:V1.

RTAStarYazidi@a418fc Chose NoOp

Pulled Node:Node@efb846- vertex-1.0 supplies:0.0 newDepth:0.0

Expanding Node:Node@efb846- v1.0 supplies:0.0 Depth:0.0

Create Node:Node@84aee7- vertex-2.0 supplies:2.0 newDepth:Infinity

Create Node:Node@9ee92- vertex-3.0 supplies:-2.0 newDepth:Infinity

Create NoOp Node:Node@f39991- vertex-1.0 supplies:-1.0 newDepth:Infinity

Pulled Node:Node@9ee92- vertex-3.0 supplies:-2.0 newDepth:Infinity

Pulled Node:Node@84aee7- vertex-2.0 supplies:2.0 newDepth:Infinity

Pulled Node:Node@f39991- vertex-1.0 supplies:-1.0 newDepth:Infinity

No Solution Found.

3) (Game trees):

Consider a 3-person game (A, B, C) with complete information and deterministic environment.

Suppose A has 5 possible moved. Provide a game tree example that shows that for each of the following set of rules, A will have a different optimal move:

a) Each agent out for itself, and they cannot communicate.

b) B and C may agree on a deal if it is beneficial to both of them

c) B and C may agree on a deal if it is nebeficial to both, but C is allowed to violate it (but B does not know that).

d) A and B may reach a deal, if it is beneficial to both of them

e) As in a, but in case of tie-breaking do a move that is worse for A.

ANSWER: The Rhombus Form **above** the graph shows how much each player will get in each of the options as (A, B, C).   
The Rhombus Form **below** the graph shows which option A will choose.



4) (Game-tree search - alpha-beta pruning)

a) Give an example of a 3-ply game tree (branching factor 2) where alpha-beta pruning saving is maximal. How many nodes are pruned?

Answer: A 3-ply game tree with branching factor 2 has 8 terminal nodes. We will let the values be integers from 1 through 8. Ordering them so that the worst move is examined first will guarantee **no pruning** (0 pruning). The reverse order will guarantee optimal pruning, We will call moves by A: A1 and A2. (4 terminal nodes are pruned)  


b) Suppose that we know ahead of time that the terminal values can only be integers between -10 and 10. Is there a case where alpha-beta can save even more search than the best case above (show it, or prove that it cannot help).

Answer: Possibly, if the maximal value (10) is reached by A1 then we can prune everything else.



c) Provide an example of a 2-ply + 2 chance nodes level game tree where one can apply an adapted alpha-beta to prune some nodes, and a similar example where changing the distribution on the chance node edges results in no savings for pruning.

ANSWER: Can only be done if values are bounded. Assume bounds are   
[-10, 10].

  
  
That is, the score of A2 cannot be higher than 4, so no point in looking further, MAX must play A1. However, if the probabilities of the chance node after A2 are (right node: 0.1, left node: 0.9) then it is still possible for the value of A2 to be higher than 5, so cannot prune.

5) (Propositional logic)

For each of the following sentences, determine whether they are Satisfiable, unsatisfiable, and/or valid. For each sentence that determine the number of models (over the propositional variables in the sentence). In case b, also trace the run of the DPLL algorithm for satisfiability with this formula as input (i.e. explain any recursive calls and cases encountered).

a) (A and (A -> B) and (B -> C)) -> C

ANSWER: equivalent to: c or 'A or ('B and A) or ('C and B)

Valid, So 8 models

b) (A or not B or D) and (not A or B) and (not A or not B or D) and (A or not B)

ANSWER: Satisfiable, 2 models: D=true and either A=False, B=True, or A=True, B=False. With DPLL, D appears only as positive, so set D=True, satisfying the first and 3rd clause. Now this cannot be repeated and there is no unit clause. So try first A=True and call recursively. This satisfies the last clause and leaves only (False or B). Now can set B=True and return success. (Would also work if we chose A=False, but with a different assignment).

c) (A or B or C or D or E or F)

ANSWER: Satisfiable. Only 1 out of 2^6 assignments is not a model, so 63 models.

d) (A and not A) and (Not B)

ANSWER: Not Satisfiable.

e) (A and (A -> B) and (B -> C)) -> not C

ANSWER: Satisfiable. True if C=False (4 models). The left-hand part negated, under C=True, is equivalent to: not A or (A and not B) which has 3 more models not overlapping the other 4, so total 7 models.   
The only non-satisfying assignment is A=B=C=True.

f) not ((A and (A -> B) and (B -> C)) -> C)

ANSWER: Negation of a valid sentence (a), so not satisfiable.

g) A -> not A

ANSWER: Equal to: !a

Satisfiable, 1 model.

6)===================================