Tutorial Letter 202/2/2018

Techniques of Artificial Intelligence

COS3751

Semester 2

School of Computing

IMPORTANT INFORMATION

This tutorial letter contains model solutions for assignment 02

BAR CODE

Define Tomorrow.

UNISA university of south africa

ASSIGNMENT 02
Solution
UNIQUE ASSIGNMENT NUMBER: 709589

Study material: Chapters 5, 6, 7, and 8. You may skip sections 5.5, 5.6, and 5.7, 7.6 and 8.4.

Question 1

(1.1) Clearly explain what an evaluation function is, and why it is used during adversarial searches.

Given a terminal or non-terminal state s (i.e. a board position) and a player p, an evaluation function returns a numerical value of the strength of state s for player p. An evaluation function can be used to choose the best moves for a player, which branches of a search tree to prune, or to provide a kind of utility value when the look-ahead depth does not reach a terminal node.

(1.2) Is the ideal strategy only available if we have perfect information? Explain your answer.

No. Agents can still get the ideal strategy when playing without all the information. This means the ideal strategy includes the notion of limited information, however, an agent that has perfect information will most likely outperform an agent that has limited/imperfect information for the same problem.

(1.3) Explain how forward pruning works. Provide at least one approach to forward pruning in your explanation, as well as a problem that may be encountered with forward pruning.

Forward pruning means that some nodes are pruned without even considering them. Beam search [others are also considered] only considers a sample of the best moves at each ply, but this may lead to the best move *being pruned away.*

(1.4) Does the order in which nodes are examined in minimax matter? Explain your answer.

Only if we prune. Otherwise minimax is an exhaustive search (for the parameters such as depth – plys – it is configured for) and node orders do not matter.

Question 2

Consider Figure 1 and answer the questions that follow. (The utility values of the leaf nodes are provided below the leaf nodes.)

(2.1) Provide the minimax values for all the nodes.

A = 7, B = 7, C = 6, E = 9, G = 6

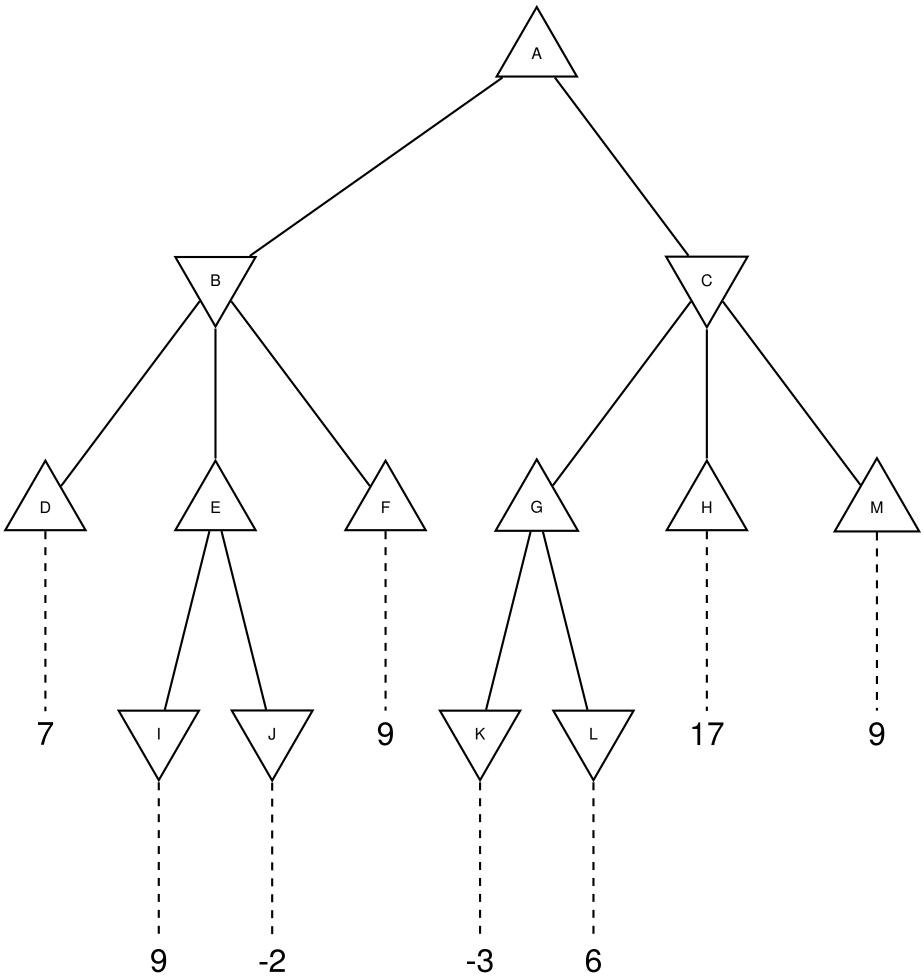
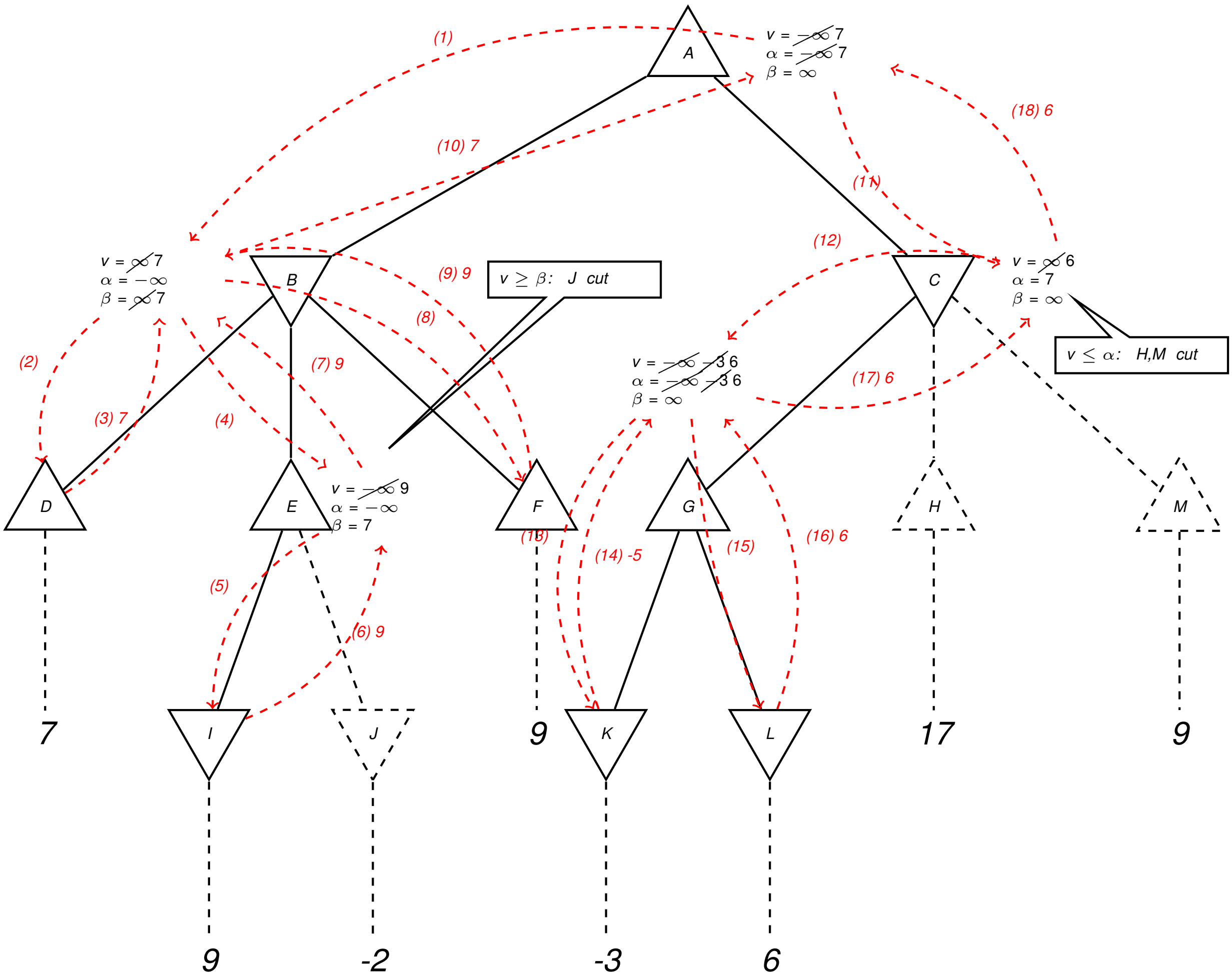


Figure 1: Minimax, alpha/beta

(2.2) Which move should MAX make? Explain your answer.

B. The utility value for MAX for move B is higher than that of move C.

(2.3) Write down the α/β values for all the nodes (except the leaf nodes) if alpha/beta pruning is applied to the tree.



The final values are: $A = (7, \infty), B = (-\infty, 7), E = (-\infty, 7), C = (7, \infty), G = (6, \infty)$

(2.4) Write down which nodes were cut and what type of cut was made in each case (alpha, or beta).

J was beta-cut, H and M were alpha-cut.

Question 3

Consider the *subtraction* game: two players (A and B) take turns removing items from a heap (just one heap). Each player may remove either one, two, or three items from the heap. The heap starts off with 12 items, and player A moves first. The objective of the game is to be the last player to remove items from the heap. That is, if it is your turn to move, and the heap is empty, you've lost the game[1].

The Initial state for the game is $(A, 12)$ indicating that it is A's turn to move and there are 12 items on the heap. In general then, a state is represented as (P, n) where $P \in \{A, B\}$, and $0 \leq n \leq 12$.

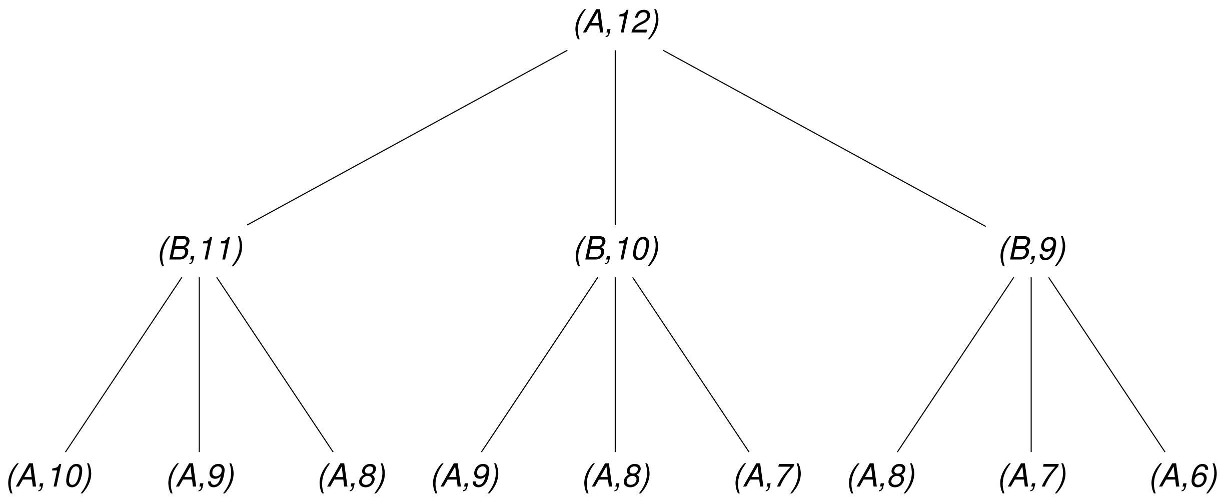
The evaluation function for non-terminal nodes is a simple threshold function defined as:

$$eval(S) = \begin{cases} -1 & \text{if } S.n \text{ mod } (k+1) = 0, \\ 1 & \text{otherwise} \end{cases}$$

[1]visit https://youtu.be/aonCsvi0LKc to see how the game is played with 21 items.

S.*n* refers to the number of items left in the heap in state *S*, and *mod* is the integer modulo function. *k* is the maximum number of items that may be removed on a turn, so in this case $k = 3$.

(3.1) Draw the entire game tree, starting from the initial state, down to depth two (the initial state is at depth 0), and provide the evaluation for each state at depth 2.



The values of the states at depth 2 are as follows:
$eval((A,10)) = 1$ *since* $(10 \ mod \ (3+1) = 10 \ mod \ 4 = 2$.
$eval((A,9)) = 1$ *since* $9 \ mod \ 4 = 1$.
$eval((A,8)) = -1$ *since* $8 \ mod \ 4 = 0$.
$eval((A,7)) = 1$ *since* $7 \ mod \ 4 = 3$.
$eval((A,6)) = 1$ *since* $6 \ mod \ 4 = 2$.

(3.2) Using the minimax algorithm, provide the backed-up values for states at depth 1 and 0. (**Hint**: Since you don't have the entire game tree, and thus no terminal states, you cannot use a utility value to calculate the backed up values. However, you do have an evaluation function.)

The values of states on level 2 (which is a max level) are given in the answer to 3.1.
At depth 1 (a min level): $(B,11)$ *gets the value* $MIN(eval((A,10)), eval((A,9)), eval((A,8)) = MIN(1,1,-1) = -1$.
$(B,10)$ *gets the value* $MIN(eval((A,9)), eval((A,8)), eval((A,7)) = MIN(1,-1,1) = -1$.
$(B,9)$ *gets the value* $MIN(eval((A,8)), eval((A,7)), eval((A,6)) = MIN(-1,1,1) = -1$.
Finally, at depth 0 (a max level): $(A,12)$ *gets the value* $MAX(-1,-1,-1) = -1$.

(3.3) Which node(s) would not have been evaluated at depth 2 if alpha/beta pruning was employed?

Below (B,10): (A,7). Below (B,10): (A,7) and (A,6).

Question 4

Answer the following questions on Constraint Satisfaction Problems (CSPs).

(4.1) Define the Least Constraining Value (LCV) heuristic.

The heuristic prefers values that rule out the fewest choices for neighbours (it tries not to limit the number of choices that remain for neighbours).

(4.2) Explain why establishing strong k-consistency is a problem.

This means we must show that the graph is k-consistent, $k-1$-consistent and so on. This can only be done in exponential time which makes large problems intractable.

(4.3) Define the *degree* heuristic.

This heuristic attempts to reduce the branching factor on future choices.

(4.4) If no legal assignments for a variable remain during a solution to a CSP, does it mean that the algorithm will be able to find a solution by simply backtracking? Explain your answer.
No. It may very well be that there is no answer (there is no form of consistency for the problem given the constraints, resulting in no solution).

(4.5) Explain what forward checking for a CSP is.

It is a domain reduction technique which establishes arc *consistency. (Or: It removes values from the domain of the neighbours of the variable for which the forward checking is being done.)*

Question 5

CSPs are especially useful when trying to solve scheduling problems.
Consider the problem of determining how to assign aircraft in a fleet to particular flights. A flight (or a leg) is simply a scheduled transference of passengers from a departing airport to a destination airport (with no stops inbetween). In order to transfer passengers between the source and destination airport a flight needs an aircraft.
Additionally, aircraft have a minimum turn-around time of 30 minutes. That is, an aircraft cannot arrive, and then simply take off again. It has to taxi to the gate, passengers should debark, the aircraft should be cleaned, and the new passengers should embark. Only then can the aircraft taxi to the assigned runway and then take-off.
The airline in question operates a small fleet of short-haul aircraft, and all flights are only for 9 or fewer passengers.
Consider the following flight schedule:

Flight	Departs	Lands
QQ002	9:15	10:45
QQ002	14:00	14:45
QQ004	12:15	13:15
QQ008	10:45	11:45
QQ016	09:30	10:15
QQ032	11:15	13:30
QQ064	13:00	13:45
QQ128	13:15	14:15
QQ256	13:30	14:45
QQ512	10:00	10:45
QQ512	14:00	14:45

You can assume that departure happens from the same airport (i.e. you don't have to worry about the location of the aircraft).
The airline owns several Cessna 208 Grand Caravans (208B), which are:

1. XAX-344,
2. XAX-254,
3. XAX-983,
4. XAX-124
5. XAX-888

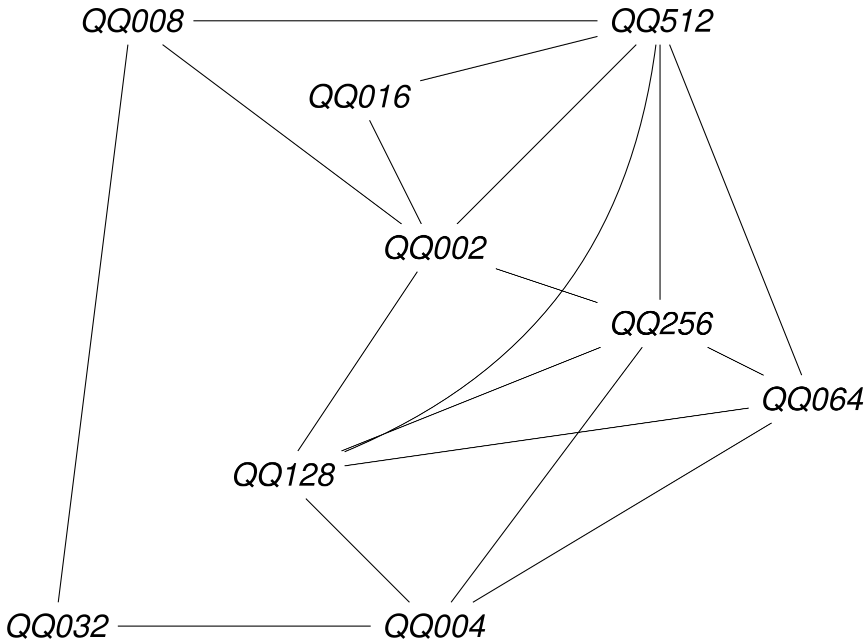
(5.1) The first step when representing a problem as a CSP is to define the variables. The variables are those elements in the problem that get assigned something else from the problem. Read the above problem description carefully to determine which elements from the problem are variables. Now provide these variables for the problem. Remember to use the correct notation!
Hint: In most cases the variables are those elements from the problem which will require the use of a limited item available.
The variables are simply the flights: X = {QQ002, QQ004, QQ008, QQ016, QQ032, QQ064, QQ128, QQ016,QQ128}.

(5.2) Once the variables have been defined, we proceed to defining the domain for each variable. That is, the 'value' that each variable can take. Define the domain for each variable in the CSP.
The domain for each variable will be the flight that it can take on. D_X = { *XAX-344, XAX-254, XAX-983, XAX-124* }

(5.3) The constraints determine the 'restrictions' placed on variables. Define the constraints for the variables in the CSP.

1. $QQ002 \neq QQ008$
2. $QQ002 \neq QQ016$
3. $QQ002 \neq QQ128$
4. $QQ002 \neq QQ256$
5. $QQ002 \neq QQ512$
6. $QQ004 \neq QQ032$
7. $QQ004 \neq QQ064$
8. $QQ004 \neq QQ128$
9. $QQ004 \neq QQ256$
10. $QQ008 \neq QQ032$
11. $QQ008 \neq QQ512$
12. $QQ016 \neq QQ512$
13. $QQ064 \neq QQ128$
14. $QQ064 \neq QQ256$
15. $QQ064 \neq QQ512$
16. $QQ128 \neq QQ256$
17. $QQ128 \neq QQ512$
18. $QQ256 \neq QQ512$

(5.4) Provide the constraint graph for this problem.



(5.5) Provide the solution to the problem. Use the Minimum Remaining Values (MRV) heuristic, and at each step establish arc-consistency for the variables. Show how the solution is calculated in a step by step fashion. Show the variables and their assigned values as your final answer!

Using MRV we easily choose any of the variables, and we can easily assign any aircraft to them. Choose QQ002 = XAX-344.

- $D_{QQ004,QQ032,QQ064}$ = { *XAX-344, XAX-254, XAX-983, XAX-124, XAX-888* }
- $D_{QQ008,QQ016,QQ128,QQ256,QQ512}$ = { *XAX-254, XAX-983, XAX-124, XAX-888* }

Using MRV we now select any of the variables that had their domain reduced from the previous step. Choose QQ008 = XAX-254.

- $D_{QQ004,QQ064}$ = { *XAX-344, XAX-254, XAX-983, XAX-124, XAX-888* }
- $D_{QQ016,QQ128,QQ256}$ = { *XAX-254, XAX-983, XAX-124, XAX-888* }
- D_{QQ512} = { *XAX-983, XAX-124, XAX-888* }

We choose QQ512 since it now has the least remaining values. Choose QQ512 = XAX-983.

- D_{QQ004} = { *XAX-344, XAX-254, XAX-983, XAX-124, XAX-888* }
- $D_{QQ016,QQ256}$ = { *XAX-254, XAX-124, XAX-888* }
- D_{QQ032} = { *XAX-344, XAX-983, XAX-124, XAX-888* }
- D_{QQ064} = { *XAX-344, XAX-254, XAX-124, XAX-888* }

- D_{QQ128} = { *XAX-254, XAX-983, XAX-124, XAX-888* }

We choose QQ016 since it now has the least remaining values (along with QQ256). Choose QQ016 = XAX-254.

- D_{QQ004} = { *XAX-344, XAX-254, XAX-983, XAX-124, XAX-888* }
- D_{QQ032} = { *XAX-344, XAX-983, XAX-124, XAX-888* }
- D_{QQ064} = { *XAX-344, XAX-254, XAX-124, XAX-888* }
- D_{QQ128} = { *XAX-254, XAX-983, XAX-124, XAX-888* }

We choose QQ256 since it now has the least remaining values. Choose QQ256 = XAX-254.

- D_{QQ004} = { *XAX-344, XAX-983, XAX-124, XAX-888* }
- D_{QQ032} = { *XAX-344, XAX-983, XAX-124, XAX-888* }
- D_{QQ064} = { *XAX-344, XAX-124, XAX-888* }
- D_{QQ128} = { *XAX-983, XAX-124, XAX-888* }

We choose QQ032 since it now has the least remaining values (along with QQ064). Choose QQ032 = XAX-344.

- D_{QQ004} = { *XAX-983, XAX-124, XAX-888* }
- D_{QQ064} = { *XAX-344, XAX-124, XAX-888* }
- D_{QQ128} = { *XAX-983, XAX-124, XAX-888* }

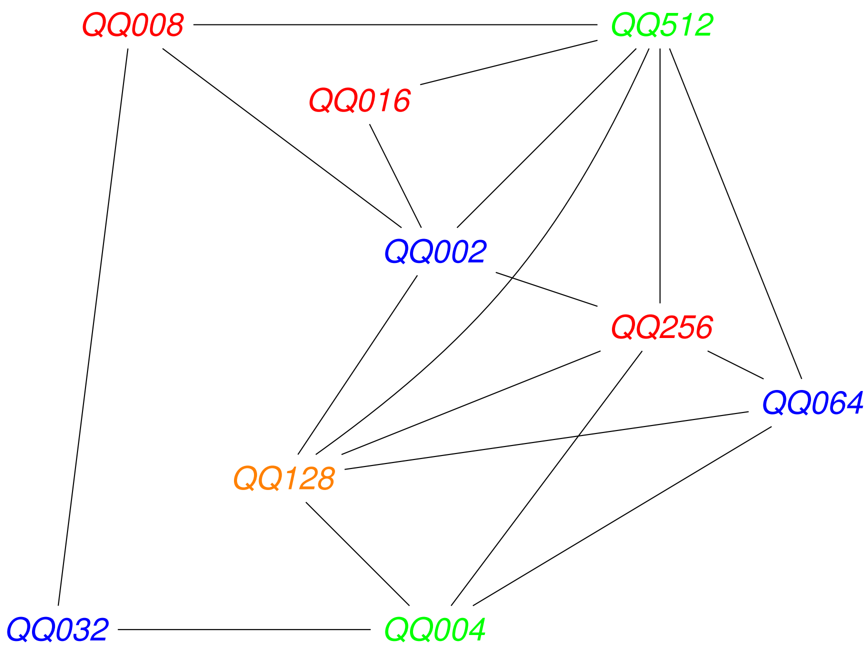
We choose QQ004. Choose QQ004 = XAX-983.

- D_{QQ064} = { *XAX-344, XAX-124, XAX-888* }
- D_{QQ128} = { *XAX-124, XAX-888* }

We choose QQ128 since it now has the least remaining values. Choose QQ128 = XAX-124.

- D_{QQ064} = { *XAX-344, XAX-888* }

We choose QQ064 since it now has the least remaining values. Choose Q064 = XAX-344.
The problem is thus solved. The coloured graph is provided for verification.



Flight	*Plane*
QQ002	*XAX-344*
QQ004	*XAX-983*
QQ008	*XAX-254*
QQ016	*XAX-254*
QQ032	*XAX-344*
QQ064	*XAX-344*
QQ128	*XAX-124*
QQ256	*XAX-254*
QQ512	*XAX-983*

Question 6

Consider the following statements from a knowledge base *KB*:

a. $(P \land Q) \Rightarrow R$

b. $T \Rightarrow Q$

c. $W \Rightarrow P$

d. $\neg R$

Prove that $KB \models (W \Rightarrow \neg T)$ using resolution refutation. (Hint: FIrst convert the statements in *KB* and the negation of the goal to clause form. Note that $\neg(W \Rightarrow \neg T) \equiv \neg(\neg W \lor \neg T) \equiv W \land T$.)

Here is a short example of what your resolution proof should look like:

1.	$\neg A \vee B$	(premise)
2.	A	(premise)
3.	$\neg B$	(negation of goal)
4.	B	(1 & 2)
5.	$\emptyset$	(3 & 4)

When using refutation, we prove by contradiction. That is we assume that which we are asked is not true, and try to find a contradiction. Using refutation this simply means adding $\neg(W \Rightarrow \neg T)$ to the KB, repeatedly applying resolution, and if the closure contains the empty clause we have a contradiction. The hint provided shows that the negation of the goal is equivalent to $W \wedge T$, thus we add W and T to the clausal form of KB, and use resolution to try and derive the empty clause.

1.	$\neg P \vee \neg Q \vee R$	*(premise)*
2.	$\neg T \vee Q$	*(premise)*
3.	$\neg W \vee P$	*(premise)*
4.	$\neg R$	*(premise)*
5.	W	*(negation of goal)*
6.	T	*(negation of goal)*
7.	$\neg P \vee \neg Q$	*(1 and 4)*
8.	Q	*(2 and 6)*
9.	$\neg P$	*(7 and 8)*
10.	P	*(5 and 3)*
11.	$\emptyset$	*(9 and 10)*

We arrive at a contradiction, and thus we can conclude that KB $\models (W \Rightarrow \neg T)$.

Question 7

Consider a vocabulary with the following symbols:

$Customer(p1, p2)$:	Predicate. Person $p1$ is a customer of person $p2$.
$Boss(p1, p2)$:	Predicate. Person $p1$ is a boss of person $p2$.
$Doctor(p)$:	Predicate. Person p is a doctor.
$Surgeon(p)$:	Predicate. Person p is a surgeon.
$Lawyer(p)$:	Predicate. Person p is a lawyer.
$Actor(p)$:	Predicate. Person p is an actor.
$Emily, Joe$:	Constants denoting people.

Use these symbols to write the following assertions in first-order logic:

a. Emily is either a surgeon or a lawyer (but not both).

b. All surgeons are doctors.

c. Joe does not have a lawyer (i.e. he is not the customer of any lawyer).

d. There exists a lawyer all of whose customers are doctors.

e. Every surgeon has a lawyer.

The translation is straighforward using the vocabulary provided.

a. $(Surgeon(Emily) \lor Lawyer(Emily)) \land \neg(Surgeon(Emily) \lor Lawyer(Emily))$

b. $\forall p\ (Surgeon(p) \Rightarrow Doctor(p))$

c. $\neg\exists p\ (Lawyer(p) \land Customer(Joe, p))$

d. $\exists p\ (Lawyer(p) \land \forall q\ (Customer(q, p) \Rightarrow Doctor(q)))$

e. $\forall p\ (Surgeon(p) \Rightarrow \exists q\ (Lawyer(q) \land Customer(q, p)))$