1) In one of the lisp lectures we looked at the logical connective NAND. NAND has the property that it can replace all other logical connectives, although the expressions get very complex. Another connective which has this property is NOR.

The truth table for NOR is:

<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>p NOR q</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

Use the rewriting rules below to define a lisp function which transforms an expression with AND, OR and NOT to an expression with just NOR.

(not p) → (nor p p)
(and p q) → (nor (nor p p) (nor q q))
(or p q) → (nor (nor p q) (nor p p))

You may define auxiliary functions if you want. [5]
2) Suppose we have a graph with nodes S, A, B, and C, where S is the start and G the goal node. The distances between connected nodes are given in this table:

<table>
<thead>
<tr>
<th></th>
<th>S</th>
<th>A</th>
<th>B</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>4</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>4</td>
<td>1</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>6</td>
<td>1</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>6</td>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Perform a search with method A:
- Draw the tree with the g and h values for all nodes.
- Show in what order the nodes are examined and
- Discuss whether the heuristic function is optimistic and/or monotone and how this affects the search.

Do this for the three cases when the estimated remaining distances are:

a) \( h(A) = 4 \), \( h(B) = 1 \)

**Diagram:**

```
  S
  /\  
  /  \  
 A   B
  \  / \
   4+4 6+1
  /\    /\  
 B   G A   G
  \  /   \  /   \
  5+1 10 7+4 10
   \     \
    G
     9
h(A) = 4, h(B) = 1
Optimistic, but not monotone
```
b) \( h(A) = 9, \ h(B) = 6 \)

\[
\begin{array}{c}
\text{S} \\
\text{A} \\
\text{B} \\
\text{A} \\
\text{G}
\end{array}
\]

\[
\begin{array}{c}
4 + 9 \\
6 + 6 \\
7 + 9 \\
10
\end{array}
\]

\( h(A) = 9, \ h(B) = 6 \)

Not optimistic, and therefore not monotone

c) \( h(A) = 4, \ h(B) = 3 \)

\[
\begin{array}{c}
\text{S} \\
\text{A} \\
\text{B} \\
\text{B} \\
\text{G} \\
\text{G}
\end{array}
\]

\[
\begin{array}{c}
4 + 4 \\
6 + 3 \\
5 + 3 \\
10 \\
9
\end{array}
\]

\( h(A) = 4, \ h(B) = 3 \)

Optimistic and monotone

If \( h(n) \) is monotone, when a node is found it is via the shortest path, so the node’s \( g \)-value is known.

If \( h(n) \) is optimistic the shortest path will be found.
In a Bayesian network we have the following nodes:

- $x_1$: smoking
- $x_2$: visit_to_Africa
- $x_3$: lung_cancer
- $x_4$: bronchitis
- $x_5$: tuberculosis
- $x_6$: coughing
- $x_7$: positive_chest_x_ray

$x_1$ can cause $x_3$ and $x_4$
$x_3$ can cause $x_6$
$x_4$ can cause $x_6$ and $x_7$
$x_2$ can cause $x_5$
$x_5$ can cause $x_7$

a) Draw a graph which shows the dependencies.

![Bayesian Network Diagram]

Suppose we learn that $x_6$ is true.

b) How does this affect the probabilities of the other nodes?

You can reason in a number of ways, but I prefer this:

b) $x_6$ is true:
x3, x4 and x1 become more likely by abduction, and x7 due to x4
x5 and x2 are hard to judge, can go up or down

Then we learn that $x_7$ is true.

c) How does this affect the probabilities of the other nodes?

c) $x_7$ is also true:
x4 increases further, while x3 decreases

Then we learn that $x_2$ is true.

d) How does this affect the probabilities of the other nodes? [4]

d) $x_2$ is also true
x5 becomes very likely,
and x4 decreases to the same level as x3
4) In the book the following context-free rules are used to describe noun phrases:

\[
\begin{align*}
\text{noun\_phrase} & \rightarrow \text{noun} \\
\text{noun\_phrase} & \rightarrow \text{article\ noun} \\
\text{noun} & \rightarrow \text{man} \\
\text{noun} & \rightarrow \text{dog} \\
\text{article} & \rightarrow \text{a} \\
\text{article} & \rightarrow \text{the}
\end{align*}
\]

a) Extend the grammar to include preposition phrases. A preposition phrase is a preposition followed by a noun phrase. Prepositions are \textit{in} and \textit{on}. A noun phrase can now be a noun, a noun followed by a preposition phrase, an article followed by a noun, or an article followed by a noun followed by a preposition phrase.

The complete grammar (including the two nouns from b) is now:

\[
\begin{align*}
\text{noun\_phrase} & \rightarrow \text{noun} \\
\text{noun\_phrase} & \rightarrow \text{noun\ preposition\_phrase} \\
\text{noun\_phrase} & \rightarrow \text{article\ noun} \\
\text{noun\_phrase} & \rightarrow \text{article\ noun\ preposition\_phrase} \\
\text{preposition\_phrase} & \rightarrow \text{preposition\ noun\_phrase} \\
\text{noun} & \rightarrow \text{man} \\
\text{noun} & \rightarrow \text{dog} \\
\text{noun} & \rightarrow \text{house} \\
\text{noun} & \rightarrow \text{prairie} \\
\text{preposition} & \rightarrow \text{in} \\
\text{preposition} & \rightarrow \text{on} \\
\text{article} & \rightarrow \text{a} \\
\text{article} & \rightarrow \text{the}
\end{align*}
\]

b) Show also the parse tree for the phrase:

\[
\text{the\ dog\ in\ the\ house\ on\ the\ prairie}
\]

(You have to include two more nouns too.) [5]
the dog in the house on the prairie
5) We want to learn if a creature lives or dies, based on its genome. For simplicity we consider creatures with genomes of only 6 letters (each letter can be A, C, G or T). We find the following examples.

CATCAT – lives
TACCAT – dies
GATCAG – lives
ACTAAC – dies
TTTCCC – dies
GATCTA – lives
AATTGA – dies

a) Use the Candidate Elimination algorithm on this example to learn the category of living creatures. Candidate categories are strings of 6 letters from \{A,C,G,T,*\}. Use the examples in the above order, and show the sets S and G for each step.

<table>
<thead>
<tr>
<th>example</th>
<th>S</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ CATCAT</td>
<td>CATCAT</td>
<td>*****</td>
</tr>
<tr>
<td>- TACCAT</td>
<td>CATCAT</td>
<td>C*****</td>
</tr>
<tr>
<td>+ GATCAG</td>
<td><em>ATCA</em></td>
<td><strong>T</strong>*</td>
</tr>
<tr>
<td>- ACTAAC</td>
<td><em>ATCA</em></td>
<td><em>AT</em>**</td>
</tr>
<tr>
<td>- TTTCCC</td>
<td><em>ATCA</em></td>
<td><em>AT</em>**</td>
</tr>
<tr>
<td>+ GATCTA</td>
<td><em>ATC</em>*</td>
<td><em>ATC</em>*</td>
</tr>
<tr>
<td>- AATTGA</td>
<td><em>ATC</em>*</td>
<td><em>ATC</em>*</td>
</tr>
</tbody>
</table>

b) Use the ID3 algorithm to learn a decision tree for the above examples. Decisions are of the kind “What is the n:th letter?” (NOTE: normally, you would need to know how to compute information gain values. But this particular case relies only on the principle of ID3.)

ID3 chooses the question with the highest information gain. In this case, the question “what is the 1st letter?” is enough to reach a complete decision: C or G: lives, T or A: dies.
c) Explain how the characteristics of the two algorithms (CE and ID3) differ, and explain in general terms how they can come to such different conclusions.

When using patterns over \{ACGT\*\}, CE can learn only a very limited set of concepts. For instance, it cannot learn the answer that ID3 finds, because it cannot represent it. CE reaches a final conclusion based on these examples, any more data could only lead to inconsistency, not to a better answer.

ID3 can in principle learn any set of living/dying patterns. It would continue to refine itself, as long as the data itself is not inconsistent (the same creature living and dying). These 7 examples are so few that there is very little confidence that we have learned the correct concept. It might just be “by accident” that the 1st letter can distinguish these cases.

CE treats living and dying asymmetrically - everyone chose to treat “living” as positive examples and “dying” as negative examples, but you could try the reverse. ID3 treats living and dying symmetrically. Moreover, it would not be constrained to two outcomes, but could handle more than two (healthy - handicapped - dead).

d) Suppose that you had not 7 but a million measurements (which is about 25 per possible genome) and that they are inconsistent. What method would you use in that case to predict if a creature with a known genome lives or dies?

Neither CE nor ID3 can handle inconsistencies, so you need to choose another method!

In this case some statistical methods would be appropriate. You can use Bayes’ rule, but that would effectively mean the following algorithm: for a given genome XXXXXX look up all occurrences of XXXXXX in the database see how many are alive, how many are dead - from this compute the “live expectancy” percentage. This is not exactly learning, and it would not take into account “similar” genomes (whether that is good or bad is a question for biologists).

You could try other methods, such as clustering, as well.
6) What steps need to be taken to go from a problem description in natural language to a solution based on resolution-based theorem proving? [4]

1. Translate the statements into logic.
   Includes: understanding natural language, decide which predicates and constants you use, and their interpretation (e.g. on(x,y) means x is on y).
2. Normalize the formula into clauses.
   Includes: eliminate implication, move negation inwards, (rename variables and) move quantifiers outwards, skolemize, turn into conjunction of disjunctions - each disjunction is a clause.
3a. Apply resolution to prove the goal, or
3b. Negate the goal, process it as above, and prove a contradiction.
   The resolution rule: from p \lor \varphi and \neg p \lor \psi, conclude \varphi \lor \psi.
   To get the same p at both sides, you may need unification to instantiate variables.

7) Describe the architecture of a typical rule based expert system. [3]

See [http://www.it.uu.se/edu/course/homepage/ai/vt08/Expert.pdf](http://www.it.uu.se/edu/course/homepage/ai/vt08/Expert.pdf) (first slide) or Luger page 279 Figure 8.1.
Note: Match is more decision-tree based than rule based. It is certainly not a “typical rule based expert system”. Therefore an answer describing the tables in Match is incorrect. The architecture described in the figures above is more general, and applies to most expert systems including Match.

8) Explain in a few sentences the essence of the following concepts:
   a) horizon effect (in minimax search)
   b) non-monotonic logic
   c) the frame problem [3]

   If we use a fixed depth in minimax, the search doesn’t look further than that, and must then rely on the static evaluation. As a result, the player may try to postpone a loss, and prefer a big loss beyond the horizon over a small loss now. A solution is to make the search depth more flexible, and deepen the search in “volatile” situations.

   In monotonic logic, more information means more conclusions: you never loose a conclusion. In non-monotonic logic, more information may mean that you withdraw a conclusion. (As a consequence, you cannot treat conclusions as facts.)

   How do you specify, in some formalism, what does not change (after an action, or receiving new information). Specifying everything explicitly does not scale up, so there must be a way to put a “frame” around an event and nothing outside the frame is affected.