Question 1 [10 marks]

Consider the following logic program (assume there are declarations so there are no undefined predicate errors):

\[
\begin{align*}
a & : = \ w,x,d. \\
a & : = \ y. \\
w & : = \ z. \\
w & : = \ m. \\
z. \\
x & : = \ m. \\
y & : = \ z. \\
d. \\
\end{align*}
\]

(a) [5 marks] Draw the box model for \( a \). You need to include the ports (boxes and lines/arrows), but not the port names. You need to include the names for the atoms that the boxes represent.

(b) [5 marks] Here is a (edited) trace of the query \(?- a.\) Fill in the missing (underlined) lines:

```prolog
[trace]   ?- a.
  Call: a
  Call: w

______________________________
Exit: z

______________________________
Call: x
Call: m
Fail: m

______________________________
Redo: w
Call: m
Fail: m
Fail: w

______________________________
Call: y
Call: z
Exit: z

______________________________
Exit: a
true.
```
Question 2 [12 marks]

(a) [4 marks] What does “g is not a logical consequence of KB” mean? [Copying the definition of logical consequence from your notes and putting “not” in the front will not result in many marks.]

(b) [6 marks] Consider the following (partial) derivation of the query ?w. Note that the knowledge base is not specified. Fill in the underlined missing answers.

<table>
<thead>
<tr>
<th>Answer clause</th>
<th>Clause resolved</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes :- e</td>
<td>e :- g, h.</td>
</tr>
<tr>
<td>yes :- g, h</td>
<td></td>
</tr>
<tr>
<td>yes :- k, d, h</td>
<td></td>
</tr>
<tr>
<td>yes :- m, d, h</td>
<td></td>
</tr>
<tr>
<td>yes :- d, h</td>
<td></td>
</tr>
<tr>
<td>(c)</td>
<td></td>
</tr>
<tr>
<td>(c)</td>
<td></td>
</tr>
<tr>
<td>yes :- a, h</td>
<td></td>
</tr>
</tbody>
</table>

(c) [2 marks] If the proof fails here, what can you say about the knowledge base?

Question 3 [10 marks]

In assignment 1, we wrote a program where the solution was:

```
-- myapply lst sub where sub is a list of (x,y) pairs, replaces each occurrence of x by y in lst.
myapply :: Eq t => [t] -> [(t, t)] -> [t]
myapply [] _ = []
myapply (h:t) sub = app h sub : myapply t sub
  where
    -- app e sub gives the value e is replaced by according to sub
    app e [] = e
    app e ((x,y):r)
      | e==x = y
      | otherwise = app e r
```

The analogous Prolog program `myapply(Lst, Sub, Res)` is true when `Lst` is a list, `Sub` is a list of `(X, Y)` pairs, and `Res` is the result of replacing each `X` by `Y` in `Lst`. It should have the following behaviour:

?- myapply([a,b,c,d,e,c], [(a,f), (c,3), (g,7)], R).
R = [f, b, 3, d, e, 3].

?- myapply([z,a,a,z], [(a,z), (z,a)], R).
R = [a, z, z, a].

A definition of `myapply` is:

```
myapply([], _, []).
myapply([H|T], Sub, [H1|T1]) :-
  app(H, Sub, H1),
  myapply(T, Sub, T1).
```

(a) [3 marks] What is the first answer to the query
myapply([b,a,a,b], S, [c,c|R]).

(b) [7 marks] Implement app so it works with myapply. The only predefined predicate you may use is dif(X, Y) that is true when X and Y are different.

Question 4 [10 marks]

A binary search tree is a useful definition of a set. Suppose a set in Prolog is defined by the constant empty, denoting the empty set, and the term set(E, LS, RS) which denotes the set where E is a member of the set, LS is the set containing the members less than E and RS is the set of members greater than E.

The set \{2, 7, 9, 11\} can thus be represented as

\[
\text{set}(7, \text{set}(2, \text{empty}, \text{empty}), \text{set}(9, \text{empty}, \text{set}(11, \text{empty}, \text{empty})))
\]

Consider the following Prolog code:

\[
\begin{align*}
\text{memb}(E, \text{set}(E, , ,)). \\
\text{memb}(V, \text{set}(E, LT, ,)) :- \\
\quad V \#< E, \\
\quad \text{memb}(V, LT). \\
\text{memb}(V, \text{set}(E, , RT)) :- \\
\quad E \#< V, \\
\quad \text{memb}(V, RT).
\end{align*}
\]

where #< is an infix binary predicate between integers representing “less than”.

(a) [3 marks] What is the first result of the following query?

?- memb(9,S),memb(7,S).

(b) [7 marks] Implement the following relation in Prolog:

(The only predicate you can use that you do not define is #<.)

\%
\text{insert}(E, S, S1) \text{ is true if } S1 \text{ is a set containing } E \text{ and the members of set } S