Assignment Four: Logic Programming Introduction
Solution

Question One

(a) The query is `?- assignment(A, september, D).`
Here are the results from the query:

```
$ swipl
Welcome to SWI-Prolog (Multi-threaded, 64 bits, Version 7.2.3)
Copyright (c) 1990-2015 University of Amsterdam, VU Amsterdam
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

?- [cs312_2018].
true.

?- assignment(A, september, D).
A = as1,
D = 13 ;
A = as2,
D = 20 ;
A = as3,
D = 30.
```

(b) Query is `?- ta(cs312,2018,P), email(P,E).`
Just ask of the KB to get the answer. Note that this provides extra information, namely \( P \).

(c) There can’t be a conjunctive query, because it needs to check if the two variables refer to different people. We could write a rule such as the following where `dif(X,Y)` is true if \( X \) and \( Y \) refer to different people.

```
?- office_hour(P1,Day,S1,F1), office_hour(P2,Day,S2,F2),
dif(P1, P2), name(P1,FN1,LL1), name(P2,FN2,LL2).
```

(I used `dif` which is a built-in predicate that true when the symbols are different – which, in general, does not imply the individuals are different). You could write your own facts about `dif` for the given constants.

(d) You can’t define this given the language defined so far. There may be a TA called `rob` who is not listed and doesn’t have any office hours. All of the given facts would be true, and yet the answer would be
different. You could define a predicate such as \( \text{ta} \_\text{with\_no\_office\_hours}(\text{rob}) \) to indicate that \( \text{rob} \) is a TA with no office hours.

(c) You need to know:
- When classes start
- What days of the week classes are on
- Which dates are holidays
- When classes end

**Question Two**


Figure 1 shows the interpretation for the symbols in (d).

![Figure 1: The Plumbing Domain](image)

**Question Three**

(a) The following table gives the atoms added and the clause responsible for one possible sequence of clauses selected.
Atom Added | Clause
--------- | ---------
outgrabe   | outgrabe.
manxome   | manxome.
toves      | toves ← manxome.
vorpal     | vorpal ← manxome.
gyre       | gyre ← manxome.
gimble     | gimble ← outgrabe.
wabe       | wabe.
slithy     | slithy ← gyre, gimble, wabe.

(b) For the query ? − slithy. here is one failing derivation:

<table>
<thead>
<tr>
<th>Answer clause</th>
<th>Clause to resolve</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes ← slithy</td>
<td>slithy ← toves, brillig.</td>
</tr>
<tr>
<td>yes ← toves, brillig.</td>
<td>toves ← outgrabe, vorpal.</td>
</tr>
<tr>
<td>yes ← outgrabe, vorpal, brillig.</td>
<td>outgrabe.</td>
</tr>
<tr>
<td>yes ← vorpal, brillig.</td>
<td>vorpal ← manxome.</td>
</tr>
<tr>
<td>yes ← manxome, brillig.</td>
<td>manxome.</td>
</tr>
<tr>
<td>yes ← brillig.</td>
<td>brillig ← jubjub.</td>
</tr>
<tr>
<td>yes ← jubjub.</td>
<td>FAIL</td>
</tr>
</tbody>
</table>

(c) For the query ? − slithy. Here is one successful derivation:

<table>
<thead>
<tr>
<th>Answer clause</th>
<th>Clause to resolve</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes ← slithy</td>
<td>slithy : −gyre, gimble, wabe.</td>
</tr>
<tr>
<td>yes ← gyre, gimble, wabe.</td>
<td>gyre : −manxome.</td>
</tr>
<tr>
<td>yes ← manxome, gimble, wabe.</td>
<td>manxome.</td>
</tr>
<tr>
<td>yes ← gimble, wabe.</td>
<td>gimble : −outgrabe.</td>
</tr>
<tr>
<td>yes ← outgrabe, wabe.</td>
<td>outgrabe.</td>
</tr>
<tr>
<td>yes ← wabe.</td>
<td>wabe.</td>
</tr>
<tr>
<td>yes ← .</td>
<td>SUCCESS!</td>
</tr>
</tbody>
</table>

Here is part of the trace

Call: (8) brillig ? creep
Call: (9) jubjub ? creep  (*)
Fail: (9) jubjub ? creep  (**) 
Fail: (8) brillig ? creep 
Redo: (8) toves ? creep  (***)
Call: (9) manxome ? creep
Exit: (9) manxome ? creep  (****)

(*) it is calling jujube after brillig because of the clause brillig : −jubjub.
(**) jujube has no clauses so it fails. And then brillig fails as all of its clauses fail.
(***) toves needs to find another proof, so it calls the second one. SWI Prolog is reporting redo when it executes the down arrow (from fail to call).
(****) maxome succeeds as there is a fact for it.

(d) Figure
(e) There are 4 answers. There are 2 proofs for gyre, and 2 proofs for gimble. There is a proof for slithy for each combination of these.

**Question Four**

It should not have taken more than a few hours. Most of this should have been in understanding the material, not in doing busy work. I hope it was reasonable, and you learned something.

This question took about 3 minutes.