Exercise 2.1

Which of the following pairs of terms match? Where relevant, give the variable instantiations that lead to successful matching.

```
1. bread = bread                             Yes
2. 'Bread' = bread                           No
3. 'bread' = bread                           Yes
4. Bread = bread                             Yes, Bread = bread
5. bread = sausage                           No
6. food(bread) = bread                       No
7. food(bread) = X                           Yes, X = food(bread)
8. food(X) = food(bread)                     Yes, X = bread
9. food(bread,X) = food(Y,sausage)           Yes, X = sausage, Y = bread
10.                                          food(bread,X,beer) =
   food(Y,sausage,X)                         No
11.                                          food(bread,X,beer) =
   food(Y,kahuna_burger)                     No
12.                                          food(X) = X     Yes, X =
   food(**)
13.                                          meal(food(bread),drink(beer
   )) = meal(X,Y)                            Yes, X = food(bread),
                                                  Y = drink(beer)
14.                                          meal(food(bread),X) =
   meal(X,drink(beer))                       No
```

Exercise 2.2

We are working with the following knowledge base:

```
house_elf(dobby).

witch(hermione).

witch('McGonagall').

witch(rita_skeeter).

magic(X):-house_elf(X).

magic(X):-wizard(X).

magic(X):-witch(X).
```

Which of the following queries are satisfied? Where relevant, give all the variable instantiations that lead to success.

```
1. ?- magic(house_elf).    No, Error: undefined procedure wizard/1
2. ?- wizard(harry).       No, Error: undefined procedure wizard/1
3. ?- magic(wizard).       No, Error: undefined procedure wizard/1
4. ?- magic('McGonagall'). No, Error: undefined procedure wizard/1
```

```
5. ?- magic(Hermione).      Yes, Hermione = dobby ;
                                 Error: undefined procedure wizard/1
```

Draw the search tree for the fifth query `magic(Hermione).`

```
?- magic(Hermione).

   Call: (7) magic(_G312)

   Call: (8) house_elf(_G312)

   Exit: (8) house_elf(dobby)

   Exit: (7) magic(dobby)

Hermione = dobby ;

   Redo: (7) magic(_G312)

   Error: undefined procedure wizard/1
```

Exercise 2.3

Here is a tiny lexicon and mini grammar with only one rule which defines a sentence as consisting of five words: an article, a noun, a verb, and again an article and a noun.

```
word(article,a).

word(article,every).

word(noun,criminal).

word(noun,'big kahuna burger').

word(verb,eats).

word(verb,likes).


sentence(Word1,Word2,Word3,Word4,Word5) :- word(article,Word1),

                                           word(noun,Word2),

                                           word(verb,Word3),

                                           word(article,Word4),

                                           word(noun,Word5).
```

What query do you have to pose in order to find out which sentences the grammar can generate?

```
sentence(Word1,Word2,Word3,Word4,Word5).
```

List all sentences that this grammar can generate in the order Prolog will generate them. Make sure that you understand why Prolog generates them in this order.

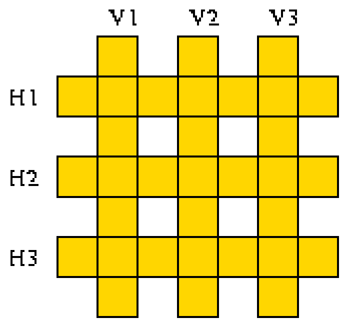
```
a criminal eats a criminal

a criminal eats a big kahuna burger

a criminal eats every criminal

a criminal eats every big kahuna burger

a criminal likes a criminal

a criminal likes a big kahuna burger

a criminal likes every criminal

a criminal likes every big kahuna burger

a big kahuna burger eats a criminal

a big kahuna burger eats a big kahuna burger

a big kahuna burger eats every criminal

a big kahuna burger eats every big kahuna burger

a big kahuna burger likes a criminal

a big kahuna burger likes a big kahuna burger

a big kahuna burger likes every criminal

a big kahuna burger likes every big kahuna burger

every criminal eats a criminal

every criminal eats a big kahuna burger

every criminal eats every criminal

every criminal eats every big kahuna burger

every criminal likes a criminal

every criminal likes a big kahuna burger

every criminal likes every criminal

every criminal likes every big kahuna burger

every big kahuna burger eats a criminal

every big kahuna burger eats a big kahuna burger
every big kahuna burger eats every criminal

every big kahuna burger eats every big kahuna burger

every big kahuna burger likes a criminal

every big kahuna burger likes a big kahuna burger

every big kahuna burger likes every criminal

every big kahuna burger likes every big kahuna burger
```

Exercise 2.4

Here are six English words:

abalone, *abandon*, *anagram*, *connect*, *elegant*, *enhance*

They are to be arranged in a crossword puzzle like fashion in the grid given below.



The following knowledge base represents a lexicon containing these words.

```
word(abalone,a,b,a,l,o,n,e).

word(abandon,a,b,a,n,d,o,n).

word(enhance,e,n,h,a,n,c,e).

word(anagram,a,n,a,g,r,a,m).

word(connect,c,o,n,n,e,c,t).

word(elegant,e,l,e,g,a,n,t).
```

Write a predicate `crosswd/6` that tells us how to fill the grid, i.e. the first three arguments should be the vertical words from left to right and the following three arguments the horizontal words from top to bottom.

```
crosswd(H1,H2,H3,V1,V2,V3) :-

    word(H1,_,A,_,B,_,C,_),

    word(H2,_,D,_,E,_,F,_),

    word(H3,_,G,_,H,_,I,_),

    word(V1,_,A,_,D,_,G,_),

    word(V2,_,B,_,E,_,H,_),

    word(V3,_,C,_,F,_,I,_).
```