Practical 7

1. The formal language *aEven* is very simple: it consists of all strings containing an even number of *a*s, and nothing else. Note that the empty string ε belongs to *aEven*. Write a DCG that generates *aEven*.

```
s --> [].

s --> a, a, s.


a --> [a].
```

2. The formal language $a^n b^{2m} c^{2m} d^n$ consists of all strings of the following form: an unbroken block of *a*s followed by an unbroken block of *b*s followed by an unbroken block of *c*s followed by an unbroken block of *d*s, such that the *a* and *d* blocks are exactly the same length, and the *c* and ~~*d*~~*b* blocks are also exactly the same length and furthermore consist of an even number of *c*s and ~~*d*~~*b*s respectively. For example, ε, *abbccd*, and *aaabbbbccccddd* all belong to $a^n b^{2m} c^{2m} d^n$. Write a DCG that generates this language.

```
s --> x.

s --> a, s, d.


x --> [].

x --> b, b, x, c, c.


a --> [a].

b --> [b].

c --> [c].

d --> [d].
```

3. The language that logicians call 'propositional logic over the propositional symbols *p*, *q*, and *r*' can be defined by the following context free grammar:

```
prop -> p

prop -> q

prop -> r

prop -> ¬ prop
```

```
prop -> (prop ∧ prop)

prop -> (prop ∨ prop)

prop -> (prop → prop)
```

Write a DCG that generates this language. Actually, because we don't know about Prolog operators yet, you will have to make a few rather clumsy looking compromises. For example, instead of getting it to recognize

```
¬ (p → q)
```

you will have to get it recognize things like

```
[not, '(', p, implies, q, ')']
```

instead. But we will learn later how to make the output nicer, so write the DCG that accepts a clumsy looking version of this language. Use `or` for ∨, and `and` for ∧.

```
prop --> [p].

prop --> [q].

prop --> [r].

prop --> not, prop.

prop --> lparen, prop, and, prop, rparen.

prop --> lparen, prop, or, prop, rparen.

prop --> lparen, prop, implies, prop, rparen.


not --> [not].

lparen --> ['('].

rparen --> [')'].

and --> [and].

or --> [or].

implies --> [implies].
```