COP4020 Programming Assignment 2

1. Write new Scheme functions `det`, `noun`, `verb`, and `adj`:

   - `det` takes a list and returns the `cdr` of the list if the first word in the list is a determiner. Otherwise, it returns an empty list `'()`.
   - `noun` takes a list and returns the `cdr` of the list if the first word in the list is a noun. Otherwise, it returns an empty list `'()`.
   - `verb` takes a list and returns the `cdr` of the list if the first word in the list is a verb. Otherwise, it returns an empty list `'()`.
   - `adj` takes a list and returns the `cdr` of the list if the first word in the list is an adjective. Otherwise, it returns an empty list `'()`.

   The vocabulary is limited to the words: `a, an, the, apple, car, dog, road, eats, occupies, rides, walks, hairy, hot, red`. Hint: you may use the `det?`, `noun?`, `verb?`, and `adj?` functions of Programming Assignment 1 to implement each of the four functions.

   Save your Scheme functions in a file named `pr2.scm`. Login with ssh to `linprog.cs.fsu.edu` and type 'scheme'. Test your Scheme functions:

   ```
   1 ]=> (load "pr2")
   2 ]=> (det '(a dog))
   ;Value: (dog)
   3 ]=> (det '(red car))
   ;Value: ()
   3 ]=> (adj '(red car))
   ;Value: (car)
   ```

2. Consider the syntax of a simple sentence that is composed of a determiner followed by a noun:

   $$\langle\text{simplesentence}\rangle \quad ::= \quad \langle\text{det}\rangle\ \langle\text{noun}\rangle$$

   Suppose we want to check if a sentence is well formed, i.e. it starts with a determiner followed by a noun. In Scheme we can test the sentence "`a dog`" with:

   ```
   1 ]=> (load "pr2")
   2 ]=> (noun (det '(a dog $)))
   ;Value: ($)
   ```

   The sentence is terminated with a special end-of-sentence symbol `$`. If the sentence is syntactically correct, the remaining `$` in the list of words is returned.

Write a function that implements `simplesentence`, such that:

```
1 ]=> (load "pr2")
2 ]=> (simplesentence '(a dog $))
;Value: ($)
3 ]=> (simplesentence '(red dog $))
;Value: #f
```

3. We can also include optional language constructs, for example

$$\langle\text{nounphrase}\rangle \quad ::= \quad [\ \langle\text{det}\rangle\ ]\ \langle\text{noun}\rangle$$

The determiner is optional, so a syntactically correct noun phrase may start with a determiner.

Write a function that implements `nounphrase`, such that:

```
1 ]=> (load "pr2")
2 ]=> (nounphrase '(a dog $))
;Value: ($)
3 ]=> (nounphrase '(red dog $))
;Value: ()
3 ]=> (nounphrase '(dog $))
;Value: ($)
```

4. Write Scheme functions for the following syntax:

$$\begin{array}{rcl}\langle\text{sentence}\rangle & ::= & \langle\text{nounphrase1}\rangle\ \langle\text{verbphrase}\rangle \\ \langle\text{nounphrase1}\rangle & ::= & [\ \langle\text{det}\rangle\ ]\ \langle\text{nounphrase2}\rangle \\ \langle\text{nounphrase2}\rangle & ::= & \langle\text{adj}\rangle\ \langle\text{nounphrase2}\rangle \\ \langle\text{nounphrase2}\rangle & ::= & \langle\text{noun}\rangle \\ \langle\text{verbphrase}\rangle & ::= & \langle\text{verb}\rangle\ [\ \langle\text{nounphrase1}\rangle\ ]\end{array}$$

Such that:

```
1 ]=> (load "pr2")
2 ]=> (sentence '(the red dog rides a hot car $))
; Value: ($)
3 ]=> (sentence '(a rides car $))
; Value: ()
```