Extra Assignment #1
Playing with Robots
version 1.0

Introduction
For your first extra assignment, you get to play with two robots. One uses two light-sensing "eyes," while another has three color sensors underneath it. Your goal is to write functions that let the first robot move towards the light and the second robot follow a green line on the ground.

The Robots
Rocinante is a circular robot about 20 cm in diameter and has a top speed of roughly 17 cm/second (0.38 mph). Rocinante has two small light sensors located very near its front. One is 5° to the left of Rocinante's nose, while the other is 5° to the right.

\[ \text{Rocinante} \]

Dilu is a circular robot about 15 cm in diameter and has a top speed of about 8.5 cm/second (0.19 mph). Dilu has three color detectors mounted under its front. One is right under the front, one is 20° to the left, and the other is 20° to the right.

\[ \text{Dilu} \]

The Sensors
Light Sensors: A light sensor tells you how much light it can see. It doesn't care where the light is coming from or what color it is. Since I haven't had that much time to work on the program, the light sensor is pretty simple. If there is a light in front of it, it will report the total brightness as some number. If the only light is behind it, it won't notice a thing and have a 0 for brightness. There are no reflections. Sorry. The amount of light that the sensor reports depends on how bright the lights are, how far away they are, and the angle between the light sensor and the light source. Naturally, a sensor facing right at the light will see more of it than a sensor looking near
the light. Please note that light sensors are not perfect! You can expect there to be a little error (at most 6%) in your readings.

Color Sensors: A color sensor tells you if the color of anything beneath it. If there is nothing right beneath it, it returns $\#f$. If there is something beneath it, it returns a sentence that tells you how much red, green, and blue that something has in it. The green line you want to follow looks like (.4 .8 .2), so it is mostly green, but it has some red and even a little blue. Keep in mind that color sensors are pretty small (about 1 cm).

**How All of This Works**

About four times a second, the robot reads its sensors and then calls a control program that you will have to write. Your program looks at the sensor data, makes a few decisions, and returns some commands. Your program will run about 400 times before the robot stops.

**What Should My Control Program Look Like?**

The details of your control program are up to you. All you have to do is make sure it has the right arguments. Your control program should take two arguments: sensors and state. sensors contains all of the sensor readings, while state contains all of the information you need to know about your robot, including its current heading (in degrees) and speed. A sample program that only tells the robot to move forward is given below:

```scheme
(define (forward sensors state)
  (command (move 2))) ; go forward
```

A program that tells the robot to swerve back and forth is:

```scheme
(define (dizzy sensors state)
  (cond
   ((>= (getheading state) 330) ; if facing down
    (command (turn 70) (speed 2))) ; turn left
   ((>= (getheading state) 30) ; if facing up
    (command (turn -70) (speed 2))) ; turn right
   (else (command (speed 2))))) ; just drive
```

Finally, a program that tells *Rocinante* to stop, turn 180°, and then continue moving when it sees light would be:

```scheme
(define (hatelight sensors state)
  (cond
   ((> (readsensor 1 sensors) 0); one eye sees something
    (command (speed 0) (turn 180)))); stop and turn
   ((> (readsensor 2 sensors) 0); other eye sees something
    (command (speed 0) (turn 180) (wait 36)))); stop & turn
   (else (command (speed 1)))); go forward
```

What Can You Do?

readsensor: This function allows you to read one of the sensors. To read the first sensor, type `(readsensor 1 sensors)`. For *Rocinante*, sensor 1 is $5^\circ$ to the right, while sensor 2 is $5^\circ$ to the left. For *Dilu*, sensor 1 is in the middle, sensor 2 is $20^\circ$ to the right, and sensor 3 is $20^\circ$ to the left.

getspeed: This function lets you find out how fast your robot is moving, or if it is moving at all. Typing `(getspeed state)` will return the current speed.

getheading: This function lets you find out where your robot is facing. It returns the heading of the robot in degrees. $0^\circ$ is to the right on the screen, $90^\circ$ is straight up, and $180^\circ$ is left. To use it, type `(getheading state)`.

turning?: This function tells you whether or not your robot is trying to turn. It returns #t if the robot is trying to turn or #f if it isn't. To use it, type `(turning? state)`.

command: command should be the return value of your function. To use it, type `(command and then begin issuing commands to the robot. The commands you may use are given below.

speed: This tells the robot how fast to move. The maximum speed for *Rocinante* is 4, while the maximum speed for *Dilu* is 2. Every time your control program runs, your robot will move a number of centimeters equal to its speed. Because I was too lazy to make it more realistic, you don't have to worry about acceleration. When you say `(speed 0)`, your robot stops. When you say `(speed 2)`, your robot will move at that speed right away.

turn: This tells the robot to turn a certain number of degrees right or left. Positive numbers move it left, while negative numbers move it right. For example, to tell the robot to start turning $37^\circ$ right, give the command `(turn -37)`.

It is important to remember that, while these robots can stop on a dime, they can't turn on one. If you tell a robot to turn $37^\circ$, it will take a little while. The robot can turn about $20^\circ$ in a second, or about $5^\circ$ every time your control function is called. Every time you tell your robot to turn, you tell it to turn more. Thus, if your robot is still trying to complete the $30^\circ$ turn you told it to do last time, and you tell it to make another $30^\circ$ turn, it will turn a total of $60^\circ$.

wait: If you want to make a $30^\circ$ turn and wait until your robot has finished it, you can say `(command (turn 30) (wait 6))`. This will start the turn and keep your control function from running six times.
noturn: This tells your robot to stop turning. It takes no arguments and looks like this:
(noturn)

How Do I Make it Work?
To run Rocinante, call the rocinate function with the name of your control program and the name of the world you want to use. There are two worlds for Rocinante. The first is onelight, which has a single red light in it. The other is manylights, which has five lights.

To run Dilu, call the dilu function with the name of your control program and the name of the world you want to use. There are three worlds for Dilu. The first one is twolines, which has, of all things, only two lines in it. The second one is polygon, which has a bunch of lines that form a loop. The third one is tricky, which has a loop as well as several lines of different colors. Don't feel the need to make Dilu go all of the way around the loop. It's probably best if you go slowly. Just make sure you drive along a few of the sides of the loop.

Example: (rocinate hatelight onelight) runs Rocinante in a world with only one light. Rocinante will drive forward unless it sees the light.

What Do I Do if Something Here Doesn't Make Sense?
If you don't understand some part of this or you find an error, send e-mail to me at ryanc@cs.berkeley.edu. Also, check the web page every so often. If we find a major goof, it will be posted there.

Don't stress over this one. I don't mind extending the deadline, especially if there are some errors. I think Rocinante is easier to do, so you might want to start with it.