CSC104 Project 2, Winter 2013
Due: Friday April 5th, 11:59 pm

You will complete two simulations, `fractal.rkt` and `contrast.rkt`, described below. Your task is to download contrast.rkt and fractal.rkt from the course website, under April 5th (right-click on them). Each of these files has comments indicating things you need to fix. The comments begin with three exclamation marks:

```
; !!! <some important work-needing instruction goes here>
```

Your job is to try to fix these, one-by-one, until you have a working simulation.

NB: Start early and leave yourself time to **ask questions** when you're stuck. Also, it's a really good idea to fix one thing at a time, and verify that it is actually fixed, before proceeding.

contrast.rkt explained

Each glowing dot in the rectangle of dots that makes up an image shines red, green, and blue light into our eyes, the mixture creating the colour we perceive. The increment contrast between high and low intensities creates visual contrast.

The simulation `contrast` experiments with making an image high contrast by increasing the intensities of light colours (those with intensity 128–255) and decreasing the intensities of dark colours (those with intensity 0–127). When you're done the simulation, you'll be able to increase the contrast of the red, green, and blue intensities independently, by hitting the "r", "g" and "b" keys, respectively. If you make all three components high enough contrast, you get an image with essentially solid colour regions.

But first, you have to either fix definitions or create **check-expect** tests for small functions used to make the colour components high contrast, and write some [question answer] pairs so that the function `toggle` will respond to keystrokes correctly.

fractal.rkt explained

Sierpinski's triangle is a classic fractal — it decomposes into three smaller sierpinski's triangles, and each of those decomposes into smaller sierpinski's. . . and so on. You can get some idea of how to create code for sierpinski's triangle online, by viewing the first video on recursion.

The simulation **fractal.rkt** allows you to increase or decrease the depth of the fractal by hitting the "up" or "down" arrow keys. If the fractal gets too deep, it will exceed the amount of screen space, so **fractal.rkt** (once you have it working) will also allow you to increase or decrease the size of the base triangle by hitting the "right" or "left" arrow keys.

Bonus possibilities — 5%

If you get **fractal.rkt** working, here are some bonus features you may add. Your grader will award up to 5% of the mark for this project as a bonus for a well-done extra feature. **NB** These bonus features will typically violate your already-working `check-expects`, so you should submit a separate file **fractalBonus.rkt** with the added features.

- Use a different fractal than sierpinski's triangle, for example the koch curve or snowflake, or another fractal of your choice.
- Randomize the color of the base triangles.
- Use a different base shape than a triangle.
- Speed up sierpinski for depths of 10 or so, using the memoization videos

What to hand in

You will submit the following files to MarkUs:

- `contrast.rkt`
- `fractal.rkt`
- (possibly) `fractalBonus.rkt`

You may work in groups of no more than 3 in preparing your project. To set up a trio or pair, one group member should log on to MarkUs and invite the other one or two. You should submit your files early and often. The first time you create a file with meaningful content, submit it. You may re-submit the same file as many times as you wish, and only the last submission is stored. A good habit is to re-submit your files each time you improve them.