1. Write a record type definition for a record type called `rgb-color`, with three fields named `red`, `green`, and `blue`. None of the fields should be mutable. Name the constructor `%make-rgb-color` to indicate that it will be only a helper procedure — the real `make-rgb-color` constructor will be defined in the next exercise.

2. Each of the components of an `rgb-color` should be an exact integer in the range from 0 up to, but not including, 256. Using `%make-rgb-color` as a helper procedure, define a three-argument constructor `make-rgb-color` that returns an `rgb-color` color record if all three of its arguments are integers in that range, and otherwise crashes with an appropriate error message. (You can use the `assert` procedure from the `(discrete utilities)` library to enforce the preconditions.)

3. Write a record type definition for a record type called `pixel`, with three fields named `x-coordinate`, `y-coordinate`, and `color`. The `x-coordinate` and `y-coordinate` fields should be mutable and so your record definition should be written in such a way that modifier procedures for them are provided automatically.

4. The constructor for the `pixel` type should ensure that its first two fields are real numbers and that its third field is a record of the `rgb-color` type. Using the technique illustrated in the second preceding exercise, arrange for the `make-pixel` constructor to enforce these preconditions.

5. Define, write, and test a Scheme procedure `move-left` that takes two arguments, a pixel and a real number, and modifies the given pixel by subtracting the given real number from its `x-coordinate`.

6. Define, write, and test a Scheme procedure `green-only` that takes a list of pixels as its argument and returns a similar list of pixels with the same `x`- and `y`-coordinates, but with each color replaced by an `rgb-color` that has the same green component as the original color but 0 for its red and blue components.

7. Define, write, and test a Scheme procedure `bluest` that takes a non-empty list of pixels as its argument and returns the one with the greatest blue component in its color field. (If two or more pixels in the list are tied for greatest blue component, the procedure may return any of them.)