MA 151: Project #1

due Tuesday October 20

Submissions

This project will be completed in groups of size 1-3, and submitted in the usual way on our class website. One member of each group should submit the code, and all group members' names should be listed in comments at the beginning of the file.

Overview

This is the first episode of a 3-part project which will focus on digital images. Here is an example image:

This is supposed to look like a thumbs-up emoji. I made it myself, obviously.

A picture like this consists of a bunch of individual squares called "pixels", each of which is either black or white. (We will introduce color pixels in Project #2.) The pixels are arranged in a grid which in this example is 16 pixels wide, and 20 pixels high.

In this project, every image is stored as a triple with type `(Int,Int,[Bool])`. The first `Int` represents the width of the image, the second is the height, and the list of Bools gives the pixels, where `True` represents a black pixel, and `False` represents a white pixel. The list simply reads the pixels from left to right across each row, from top to bottom.

For example, this image:

would be stored like this:

```
(4, 3, [True,False,True,True,False,True,False,True,False,True,True,True])
```

(Look this over carefully and be sure you agree.)

Written like this, the thumbs-up picture is pretty big. It has been included in a file `project1.hs`, where it is called `thumb`. If you load `project1.hs` into GHCi, you can evaluate `thumb` and you will see that it starts like:

```
(16,20,[True,True,True,True,True,True,True,True,True,True,True,True,True,True,True,True
```

(all those `True`s are making the first row of the border on top.

This is difficult to see in GHCi, so I have also given you a function called `display` which will show you the pixels in GHCi. Evaluate `display thumb` to see roughly what the pixels look like in your GHCi window.

Whenever you want to "see" one of our images, you will use `display`. (In Project 3, we will be able to export them to normal image files on your computer.) `display` is a weird function, and the definition might not make sense to you- it will make sense before the end of the course.

In this project you will write some simple functions to do basic operations on digital images. Throughout the whole project, every image will be stored as a triple with type `(Int,Int,[Bool])`.

Required functions

Write the following functions. Always include a type signature. Start with the file `proj1.hs` from our class website, which includes the `thumb` image and the `display` function.

- Make an image called `hImage` which looks like a capital letter "H". This is not really a function- it should have type `(Int,Int,[Bool])`. If you do `display hImage`, it should look like a nice letter "H".
- A function called `blackImage` which takes two `Int` parameters, and makes an all-black image of the given width and height.
- A function called `whiteImage` which takes two `Int` parameters, and makes an all-white image of the given width and height.
- A function called `firstRow` which takes an image and returns a `[Bool]` giving just the first row.
- A similar function called `lastRow` which gives the last row.
- Similar functions called `firstCol` and `lastCol` giving the first and last columns. The type of the answer should still be `[Bool]`. (These are maybe the hardest functions to write for this project- I suggest recursion using the height.)
- A function called `dropFirstRow` which takes an image and gives the same image but with the first row deleted.
- A similar function called `dropLastRow`.
- A function called `dropFirstRows` which takes an `Int` parameter n and an image, and deletes the first n rows.
- A similar function called `dropLastRows`.
- A function called `rowStackFirst` which takes a parameter of type `[Bool]` and an image. The first parameter is interpreted as a row, and the answer is the image obtained by "stacking" the given row on the top of the given image. (You may assume that the row parameter has the correct size (width) to be stacked onto the image parameter.)
- A similar function called `rowStackLast` which stacks the given row on the bottom of the image.
- A similar function called `rowSandwich` which stacks the given row on both the top and the bottom of the image.
- A function called `vStack` which takes two images and stacks them vertically into a big image. (The first image parameter goes on top.)
- A function called `vTile` which takes an `Int` and an image and makes a tall image by stacking the image repeatedly with itself. So `vTile 5 thumb` should make an image that consists of 5 copies of the `thumb` image stacked vertically.