BYU CS classes

Objective:

Learn the basics of the Julia Language.

Preparation:

For this lab, you should use Julia v.0.4. You may use the binary hosted in Dr. Wingate's home directory, at

```
/users/faculty/wingated/cs330/languages/julia-2e358ce975/bin
```

or you may install Julia yourself.

Deliverables:

For this lab, you will need to implement the following functions and data structures in Julia:

- `area`
- `in_shape`
- `greyscale`
- `invert`
- `count_person`
- `average_age`
- `tree_map`
- `add_last_name`
- `eye_colors`

Shapes

Define the following data structures.

```
abstract Shape

type Position
  x::Real
  y::Real
end

type Circ <: Shape
  center::Position
  radius::Real
end

type Square <: Shape
  upper_left::Position
  length::Real
end

type Rect <: Shape
  upper_left::Position
  width::Real
  height::Real
end
```

Area

```
function area(shape) ...
```

Returns the area of the given shape. where `Shape` is some concrete instance of the abstract shape type class. For this a few of the following instead of testing for class member ship it is easier to write three different functions, one for each concrete instance of shape. ie: `function area(shape::rect)`…

In_Shape

```
function in_shape(shape::Shape, position::Position) ...
```

Returns if the position is inside the shape. Assume standard math coordinates where y is positive going up. The bounds are inclusive.

Pixel

Define the Following type in your code

```
type Pixel
  r::Real
  b::Real
  g::Real
end
```

Greyscale

```
function greyscale(picture::Array{Pixel,2}) ...
```

Takes a 2d array of pixels and greys out each pixel. This is done by averaging together the r g b values and then setting r g b to the average.

Hint: Julias map works over multi-dimensional arrays

invert

```
function invert(picture::Array{Pixel,2}) ...
```

Takes a 2d array and pixels and inverts each one. Inverting is done by setting r to `255 - r` and the other two in the same manner.

Trees

Define the following types in your code

```
abstract TreeItem

type Person <: TreeItem
  name::AbstractString
  birthyear::Integer
  eyecolor::Symbol
  father::TreeItem
  mother::TreeItem
end

type Unknown <: TreeItem
end
```

Count Persons

```
function count_persons(tree)
```

Counts the number of people in the tree. Unknowns do not count as people.

Average Age

```
function average_age(tree)
```

Calculates the average age of the people in the tree. Use this years date 2016. Unknowns do not count as people.

tree map

```
function tree_map(f, tree)
```

Takes a function f and applies it to each known member of the tree. The function f takes one parameter and returns a treeItem. Tree_map will return a *new* tree. Tree_map should leave the original tree unchanged, unless f applies a mutation.

add last name

```
function add_last_name(name::AbstractString, tree)
```

Appends name to the end of the name of each member of the tree. You should use tree_map.

Hint: Julia uses * for string concatenation.

eye_colors

```
function eye_colors(tree)
```

Returns a list of all eye colors in the tree. A color can appear more than once. For grading visit the order child, father, mother.

hint: Julia arrays can be appended using the ; for example [a;b;c] will create a list with a, b, and c in it. The empty list [] is ignored.
