Names: ______________________________ ______________________________

______________________________ ______________________________

Teamwork (5)	Discussion (5)	Completeness (5)	Correctness (5)	Total (20)

Astronomical Redshift

I have many feathers to help me fly

I have a body and a head, but I'm not alive.

It is your strength which determines how far I go.

You can hold me in your hand, but I'm never thrown.

What am I?

____

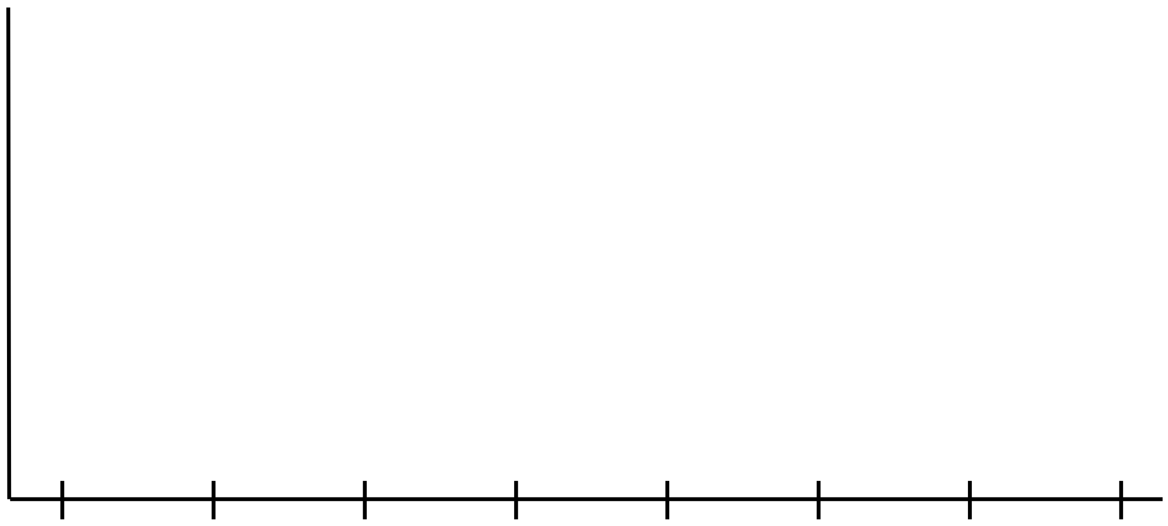
Pre-Lab Quiz

Record you team's answers as well as your reasoning and explanations.

1.
2.
3.
4.
5.

Part 1: Measuring Rest Wavelengths

1. Sketch the spectrum of Hydrogen and label the first five emission lines in the Balmer series (Hα → Hε) and record their wavelengths in the table below. Label the axis (wavelength, intensity) and let the wavelength axis span 350 to 700 nm.



Line (Transition)	Hα (3 → 2)	Hβ (4 → 2)	Hγ (5 → 2)	Hδ (6 → 2)	Hε (7 → 2)
λ_{rest} (nm)					

2. **Class Discussion** What is the point of this exercise in relation to today's lab?

Part 2: Measuring Redshifted Wavelengths

1. Identify the various Balmer and [O III] emission lines in your quasar spectrum. Then select 4 emission lines and record the observed and rest wavelengths in nanometers.

Using **Julia**, calculate the redshift value z and average the results. If any of the redshifts are significantly different than the others, you probably misidentified an emission line.

Emission Line	λ_{obs} (nm)	λ_{rest} (nm)	$\Delta\lambda = \lambda_{obs} - \lambda_{rest}$ (nm)	$z = \Delta\lambda \,/\, \lambda_{rest}$
			Average z	

Julia Programming

Open Julia and store your observed and rest wavelengths in two arrays like the following:

```
julia> w_obs  = [788, 582, 521, 490]
julia> w_rest = [656, 486, 434, 410]
```

To compute the redshift value z, we first take the difference between the observed and rest wavelengths and then divide this difference by the rest wavelengths. Make sure to include the "." in ./ for the division, otherwise Julia performs matrix division:

```
julia> dw = w_obs - w_rest
julia> z  = dw ./ w_rest              # Don't forget the "." !
```

Finally, to get the average z, we simply add the z values and divide by the number of elements:

```
julia> sum(z) / length(z)
```

2. Work out the following problems on a whiteboard then go over your work with another group. Once they agree with your calculations, show your work to the TA and have them mark below. Your TA will then provide the actual distance for use in the next problem. **Don't forget to keep track of the units!**

- Using the average redshift value, find the velocity of the quasar in km/s using the approximation $z \approx v/c$, where $c = 3 \times 10^8\,\mathrm{m/s}$ is the speed of light.
- Using Hubble's law, $v = H_0 \cdot d$, where $H_0 = 70\,\frac{\mathrm{km/s}}{\mathrm{Mpc}}$ is the Hubble constant, estimate the distance d to the Quasar in Mpc (megaparsecs).

TA	

3. Compare your distance estimate with the actual value given by the TA using the percent error formula, $\%Error = 100\% \cdot |Estimated - Actual|/Actual$

4. Did you recognize the form of Hubble's law, $v = H_0 \cdot d$? If you recognized that it is of the form $y = ax + b$, congratulations! Hubble's law is indeed a linear equation relating the recession velocity to the separation distance.

In this part, we'll fit a linear model to the data from all the groups to derive an estimate for Hubble's constant using the Julia programming language. First, we'll need to download and activate a few libraries

```
import Pkg
Pkg.add(["DataFrames", "Gadfly", "GLM"])
using DataFrames, Gadfly, GLM
```

Gadfly is a plotting library (**gadflyjl.org**). GLM stands for Generalized Linear Models and is useful for fitting linear models to data.

Next, we'll create a data frame to hold the velocities and the actual distances. Make sure to use the class values rather than the example ones given below:

```
d = [213, 367, 586, 602, 681, 940, 990]          # Distances in Mpc
v = [16, 28, 44, 45, 49, 69, 75] * 1000         # Velocities in km/s
data = DataFrame(d = d, v = v)
```

Within the GLM library is a function called `lm` which will prove useful. To learn more about this function type `?`, which will switch from **`julia>`** to **`help?>`**; then type `lm` and press return to see its documentation

```
help?> lm                    # type "?" to switch from julia> to help?>
```

From the documentation we gather that `lm` fits a linear model to the data and takes two inputs (`X, y`) and one optional input (`allowrankdeficient`). The inputs `X, y` can be either 1) a matrix and vector or 2) a formula and a data frame. Using the second option, we must specify that `X` is a formula where the velocity is dependent on the distance. We can create our linear model as follows:

```
linmod = lm(@formula(v ~ d), data)
```

Some additional output will be included, which will look something like the following:

```
             Estimate    Std. Error  t value     Pr(>|t|)    Lower 95%    Upper 95%
(Intercept)  240.829     1088.07     0.221335    0.8336      -2556.16     3037.81
d            74.0612     1.60595     46.1169     <1e-7       69.933       78.1895
```

The values under **Estimate** indicate the best-fit values. In this example, the best-fit is $v(d) = 74.0612 \cdot d + 240.829$ and our estimate for Hubble's constant is $H_0 = 74.0612\,\text{km/s/Mpc}$ (since the slope is *rise-over-run*, with [rise] = km/s and [run] = Mpc).

The column **Std. Error** indicates the uncertainty on the estimate; thus the slope and intercept are likely within the range $a = 74.0612 \pm 1.60595$ and $b = 240.829 \pm 1088.07$. From Hubble's law we would expect the intercept to equal zero, and from the above we see that an intercept of zero is well within the uncertainty of the estimate.

In our calculations we made the approximation $z \approx v/c$, which is likely to throw off our results (a couple of the spectrometers are not calibrated properly and report wavelengths that are 10 nm offset from the actual value). Record your estimate for Hubble's constant along with its uncertainty below, rounding to reasonable values.

H_0	

The remaining columns are statistical measures. The column **t value** is a measure of the significance of a non-zero value, and can be found by dividing **Estimate** by **Std. Error**. The column **Pr(>|t|)** indicates the probability of getting an estimate this far from zero if the true value is zero (where 0.8 = 80%). The last two columns indicate the 95% confidence interval.

Let's plot our data along with our best-fit linear trendline. Show your plot to the TA and have them mark below.

```
tline = DataFrame(d = 0:100:1000)          # d = 0, 100, 200, ..., 1000
tline.v = predict(linmod,  tline)          # try help?> predict
                                           # and look at the first entry
plot(
   layer(x = data.d,  y = data.v,  Geom.point),
   layer(x = tline.d, y = tline.v, Geom.line, Theme(default_color="lime")),
   Guide.xlabel("Distance (Mpc)"),
   Guide.ylabel("Velocity (km/s)")
)
```

It might take a while for the plot to generate the first time.

TA	

TA Cheat Sheet

Quasar	z	v (km/s)	D (Mpc)
J0927	.16	48,000	586
J1236	.18	54,000	681
J1400	.10	30,000	367
J1452	.16	48,000	602
J1458	.27	81,000	940
J1509	.29	87,000	990
J2105	.06	18,000	213