**Last name** _____

**First name** _____

**LARSON—MATH 310–CLASSROOM WORKSHEET 12**
**The Vec Class.**

1. Set up your CoCalc JUPYTER notebook for today's work.

   (a) Start the Chrome browser.

   (b) Go to `https://cocalc.com`

   (c) Log in.

   (d) You should see an existing Project for our class. Click on that.

   (e) Make sure you are in your Home directory (if you work in your Handouts directory, your work could get overwritten).

   (f) Click "New", then "Jupyter Notebook", then call it **310-c12**.

   (g) Make sure you have PYTHON as the *kernel*.

   **Review**

2. We coded a bare-bones version of Klein's Vec() class:

```
class Vec:
    def __init__(self, labels, function):
        self.D = labels
        self.f = function

```

3. Write a procedure `zero_vec(D)` with the following spec:
   • input: a set D
   • output: an instance of Vec representing a D-vector all of whose entries have value zero.

4. **Note:** Klein has designed his `Vec` class to promote *sparsity* (not requiring a dictionary value for every element in the vector's domain), we will have to hack out some details so that this works seamlessly.

5. Look at the docstring for `getitem` procedure. Let's make that work, and test it.

6. Look at the docstring for `setitem` procedure. Let's make that work, and test it.

7. Look at the docstring for the `add` procedure. Let's make that work, and test it.

   **New**

8. Go to your Handouts folder and copy the file "Vec_c12.py" to your Home directory. That has all our work from the last class.

9. Paste the contents into a cell of your Jupyter Notebook and run/evaluate.

10. Make a vector $\hat{v}$ to test that it's working:

```
v = Vec({1,2},{1:3})
print(v.D)
print(v.f)
```

11. Look in the Handouts folder for the *vec.py* file and the *class* definition of the vector class `Vec`. We discovered that the bracket notation for getting and setting vector elements didn't work with our basic class. Actually we can make these work. We can leverage Python's overloading. That's what these methods do.

```
    __getitem__ = getitem
    __setitem__ = setitem
```

    Add them to you Vec class and re-import, or just do this in a notebook cell and evaluate/run. Then test it.

12. Then add the following line to your Vec class and test that the "+" operator now works:

```
    __add__ = add
```

13. Look at the docstring for the `equal` procedure. Let's make that work, and test it.

14. Then add the following line to your Vec class and test that the "==" comparison operator now works:

```
    __eq__ = equal
```

15. Look at the docstring for the `scalar_mul` procedure. Let's make that work, and test it.

16. What should you add to our Vec class so that we can use an overloaded "*" with vectors? Test it.

17. Look at the docstring for the `dot` procedure. Let's make that work, and test it.

18. What should you add to our Vec class so that we can use an overloaded "*" with vectors? Test it.

19. Look at the docstring for the `neg` procedure. Let's make that work, and test it.

20. What should you add to our Vec class so that we can use an overloaded "-" with vectors? Test it.

21. What do the *methods* in the `Vec` class do?

**Getting your classwork recorded**

When you are done, before you leave class...

(a) Click the "Print" menu choice (under "File") and make a pdf of this worksheet (html is OK too).

(b) Send me an email (`clarson@vcu.edu`) with an informative header like "Math 310 - c12 worksheet attached" (so that it will be properly recorded).

(c) Remember to attach today's classroom worksheet!