**Last name** _____

**First name** _____

**LARSON—MATH 310–HOMEWORK WORKSHEET 04**
**The Vec Class.**

1. Start the Chrome browser.
2. Go to `https://cocalc.com`
3. Log in.
4. You should see an existing Project for our class. Click on that.
5. Make sure you are in your Home directory (if you work in your Handouts directory, your work could get overwritten).
6. Click "New", then "Jupyter Notebook", then call it **310-h04**.
7. Make sure you have PYTHON as the *kernel*.

**Annotate** your Jupyter notebook cells so that it is clear to anyone what problem your work corresponds to.

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

```

1. Once Python has processed this definition, you can create an instance of Vec as follows. Code and run.

```
Vec({'A','B','C'}, {'A':1})
```

2. (**Quiz 2.7.1**) 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.

3. **Test** your `zero_vec(D)` procedure. Demonstrate that it works as expected.

4. The following procedure should take a $D$-vector $\hat{v}$ as input. If $kinD$ it should return the associated function (dictionary) value and otherwise return $0$ (as described in the doc-string). Replace `pass` in the procedure description with your own code so that the procedure satisfies the spec.

```
def getitem(v,k):
    """
    Return the value of entry k in v.
    Be sure getitem(v,k) returns 0 if k is not represented in v.f.

    >>> v = Vec({'a','b','c', 'd'},{'a':2,'c':1,'d':3})
    >>> v['d']
    3
    >>> v['b']
    0
    """
    assert k in v.D
    pass
```

5. **Test** your `getitem` procedure with the docstring inputs. Demonstrate that it works as expected.

6. The following procedure should take a $D$-vector $\hat{v}$ as input and change $\hat{v}$ as described (you don't need to output anything; the default output in `None`. Your procedure will cause a change in your *environment*). Replace `pass` in the procedure description with your own code so that the procedure satisfies the spec.

```
def setitem(v,k,val):
    """
    Set the element of v with label d to be val.
    setitem(v,d,val) should set the value for key d even if d
    is not previously represented in v.f, and even if val is 0.

    >>> v = Vec({'a', 'b', 'c'}, {'b':0})
    >>> v['b'] = 5
    >>> v['b']
    5
    >>> v['a'] = 1
    >>> v['a']
    1
    >>> v['a'] = 0
    >>> v['a']
    0
    """
    assert k in v.D
    pass
```

7. **Test** your `setitem` procedure with the docstring inputs. Demonstrate that it works as expected.

8. The following procedure should take a $D$-vectors $\hat{u}$ and $\hat{v}$ as input and test if they are equal (have the same domains and their functions assign corresponding values. Remember the `Vec` class does not assume each element of the domain has a given value in its dictionary (sparse representation assumes non-provided entries are assigned 0). Use your `getitem` procedure to handle this sparse-representation issue. Replace `pass` in the procedure description with your own code so that the procedure satisfies the spec.

```
def equal(u,v):
    """
    Return true iff u is equal to v.
    Because of sparse representation, it is not enough to compare
    dictionaries

    >>> Vec({'a', 'b', 'c'}, {'a':0}) == Vec({'a', 'b', 'c'}, {'b':0})
    True
    >>> Vec({'a', 'b', 'c'}, {'a': 0}) == Vec({'a', 'b', 'c'}, {})
    True
    >>> Vec({'a', 'b', 'c'}, {}) == Vec({'a', 'b', 'c'}, {'a': 0})
    True

    Be sure that equal(u, v) checks equalities for all keys from u.f and
    v.f even if
    some keys in u.f do not exist in v.f (or vice versa)

    >>> Vec({'x','y','z'},{'y':1,'x':2}) == Vec({'x','y','z'},{'y':1,'z
    ':0})
    False
    >>> Vec({'a','b','c'}, {'a':0,'c':1}) == Vec({'a','b','c'}, {'a':0,'
    c':1,'b':4})
    False
    >>> Vec({'a','b','c'}, {'a':0,'c':1,'b':4}) == Vec({'a','b','c'}, {'
    a':0,'c':1})
    False

    The keys matter:
    >>> Vec({'a','b'},{'a':1}) == Vec({'a','b'},{'b':1})
    False

    The values matter:
    >>> Vec({'a','b'},{'a':1}) == Vec({'a','b'},{'a':2})
    False
    """
    assert u.D == v.D
    pass
```

9. **Test** your `equal` procedure with the docstring inputs. Demonstrate that it works as expected.

**Getting your homework recorded**
When you are done,

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

2. Send me an email (`clarson@vcu.edu`) with an informative header like "Math 310 - h04 worksheet attached" (so that it will be properly recorded).

3. Remember to attach your homework worksheet!