1. (a) (15 points) Write a class called `IceCream` that represents an order of ice cream. An `IceCream` starts with no scoops of ice cream, but scoops of various flavors can be added in later.

   The `IceCream` class should have the following methods with the described arguments. You should not include any additional arguments for these methods:

Method	Description
`__init__(self)`	Creates an empty `IceCream`.
`add_scoop(self, flavor, scoops)`	Adds `scoops` scoops of the given flavor to this `IceCream`.
`get_scoops(self, flavor)`	Returns the number of scoops in this `IceCream` that are of the given flavor. Returns 0 if the flavor is not in this `IceCream`.
`__eq__(self, other)`	Returns `True` if this `IceCream` has the same set of flavors as the other. Note that the number of scoops doesn't count. Returns `False` if they are not the same.

   Write your response in the box on the next page.

   (b) (5 points) In the box below, write a short program that constructs an `IceCream` objects and then adds 3 scoops of vanilla ice cream and 4 scoop of chocolate. It should then print out a message of the format
   `"Vanilla: <vanilla>, Chocolate: <chocolate>"`
   Where the values in angle brackets are found by calling the `get_scoops` method.
   You do not need to write a main method for this problem, you may write the lines of code directly in the space below.

Solution:

```
ice_cream = IceCream()
ice_cream.add_scoop('vanilla', 3)
ice_cream.add_scoop('chocolate', 2)
print('Vanilla: '
      + str(ice_cream.get_scoops('vanilla'))
      + ', Chocolate: '
      + str(ice_cream.get_scoops('chocolate')))
```

Solution:

```
class IceCream
  def __init__(self, candidates):
    self._scoops = {}

  def add_scoop(self, flavor, scoops):
    if flavor in self._scoops:
        self._scoops[flavor] += scoops
    else:
        self._scoops[flavor] = scoops

  def get_scoops(self, flavor):
    if flavor in self._scoops:
        return self._scoops[flavor]
    else:
        return 0

  def __eq__(self, other):
    return self._scoops.keys() == other._scoops.keys()
```