CSC148 Summer 2016 Quiz 02 (15 minutes)

First Name .... Last Name .... Lab room# ....

Here, it is code for a superclass **Roster**, used to keep track of records (stored as a dictionary of tuples).

```python
class Roster:
    """Represents a roster of members.
    === Attributes ===
    :type members: dict
    The members for this roster.
    """

    def __init__(self):
        """Initialize a Roster.
        """
        self.members = {}

    def add(self, member):
        """Add <member> to the roster.
        :param member: A tuple that represents a member.
        :type member: tuple
        Precondition: member[0] not in self.members len(member) > 1
        :rtype: None
        """
        self.members[member[0]] = member[1:]

    def remove(self, member):
        """Remove <member> from the roster.
        :param member: A tuple that represents a member.
        :type member: tuple
        Precondition: len(member) > 0
        :rtype: None
        """
        self.members.pop(member[0], default=None)

    def __str__(self):
        """Return a string representation of the roster.
        :rtype: str
        """
        raise NotImplementedError()
```

Also, read the context for an employee records system as follows, for which a subclass **`EmployeeRoster`** is defined in the next page.

Context: An employee roster keeps track of employees that work at a grocery store. Each employee record consists of a unique identification number, a given name, and a surname. Employees may join or leave the store, in which case they must be added or dropped from the employee roster. Assume employee data is added to the **`EmployeeRoster`** as follows: unique id, given name, and the surname. Following, is an example:

```python
my_roster.add(('945', 'Alan', 'Turing'))
```

Further, we need to be able to display all employees in the store in the following format:

```
Given name: Grace, Surname: Hopper, Employee number: 906
Given name: James, Surname: Bond, Employee number: 7
…
```

Question 1. Read the following use of **EmployeeRoster**.

As currently written, there is a bug in the **EmployeeRoster** class. For instance, following lines of code will throw an exception.

```
1) roster = EmployeeRoster()
2) roster.add((' 945', 'Alan', 'Turing'))
```

(a) State what error will be raised:

roster.add will raise an error that attribute members does not exist.

(b) How would you fix the bug?

Remove the __init__ from EmployeeRoster. (in other words, __init__ can be inherited from superclass.)

Question 2. Develop the __str__ method below.

```
from roster import Roster

class EmployeeRoster(Roster):
    """A roster of employees.
    === Public Attributes ===
    :type members: dict inherited from Roster
    """

    def __init__(self):
        """Initialize an EmployeeRoster.
        """
    pass


    # TODO: Implement __str__ for this class.
    def __str__( self):

       """
       Return a string representation of the roster.
       :rtype: str

       >>> r = EmployeeRoster()
       >>> r.add(('945', 'Alan', 'Turing'))
       >>> print(r)
       Given name: Alan, Surname: Turing, Employee number: 945

       """
       s = ""

       for id, name in self.members.items():
           s = s + "Given name: {}, Surname: {}, Employee number:
                    {}\n".format(name[0], name[1], id)
       return s
```