CS 150: Intro to Object Oriented Programming
Homework 4 Solutions

1. Exercise 10.9 - See instructor

2. (5 points) Exercise 10.4

   Solution: The __contains__ method in python's list class works under the premise of (B): it merely checks if the list contains an element that is equivalent to the give parameter, *not* that they are the exact same object. In order to test this, you needed to check a list of mutable objects (since python often makes different variables point to the same immutable object to save space, as discussed in class - see section 10.1.4, page 340 for more details).

   Here is a command line session which I used to verify this fact; note that I used the same Account class described in the book and in class. (You could have used any similar mutable class to get the same result.)

```
>>> from accountclass import Account
>>> myDebit = Account()
>>> mySavings = Account()
>>> newaccount = Account()
>>> myAssets = [mySavings, myDebit]
>>> newaccount in myAssets
True
>>> id(newaccount)
140289990858584
>>> id(myAssets[0])
140289990858512
>>> id(myAssets[1])
140289990858440
>>>
```

   ■

3. (5 points) Exercise 11.10

   Solution: Below is one way to code such a solution:

```
def index(self, value, start=0):
    if self._isEmpty( ):
        raise ValueError('OurList.index(x): x not in list')
    elif start == 0 and self._head == value:
        return 0
    else:
        return 1 + self._rest.index(value, max(start?1,0))
```

   ■

4. (5 points) Exercise 11.24

Solution:

```
def binary(n):
    if n <= 1:
        return str(n)
    else:
        return binary(n // 2) + str(n % 2)
```

■

5. (5 points) Exercise 11.17

Solution: Again, any code which worked would be acceptable for this problem. Note that for full credit, you needed to actually mutate the original list, *not* return a new list.

```
def reverse(self):
    if not self._isEmpty():
        self._rest.reverse()
        self.append(self._head)
        self.remove(self._head)
```

■