**Chapter 8 test problems**

1. Write a class named Article that models items with an ID and price. The class should support methods setID() and setPrice() that set the ID and price of an item, respectively, and methods getID() and getPrice() that return the ID and price of an item, respectively.

>>> item = Article()

>>> item.setID('199000992')

>>> item.setPrice(0.99)

>>> item.getPrice()

0.99

>>> item.getID()

'199000992'

2. Write a class named Article that models items with an ID and price. The class should have an overloaded constructor that takes as input the item’s ID and price. It should also support a method setPrice() that sets the item’s price, and methods getID() and getPrice() that return the ID and price of an item, respectively.

>>> item = Article('199000992', 0.99)

>>> item.getID()

'199000992'

>>> item.getPrice()

0.99

>>> item.setPrice(1.09)

>>> item.getPrice()

1.09

3. Write a class named PingPong that has a method next() that alternates between printing ‘PING’ and ‘PONG’ as shown below.

>>> ball = PingPong()

>>> ball.next()

PING

>>> ball.next()

PONG

>>> ball.next()

PING

>>> ball.next()

PONG

4. Write a class named Counter that models an integer counter. It should have an overloaded constructor that takes as input the initial (integer) value of the counter; if no integer is passed to the constructor, the default initial value should be 0. The class should also support the method next() that returns the value of the counter and increments it.

>>> count = Counter(1)

>>> count.next()

1

>>> count.next()

2

>>> count.next()

3

>>> count = Counter()

>>> count.next()

0

>>> count.next()

1

5. Write a class named Cursor that models an integer “cursor”. When created, the cursor object should have value 0. The class should support method curr() that returns the value of the cursor, method next() that increments the value of the cursor, and method prev() that decrements the value of the cursor. Methods prev() and next() should also return the new value of the cursor.

>>> cursor = Cursor()

>>> cursor.curr()

0

>>> cursor.next()

1

>>> cursor.next()

2

>>> cursor.next()

3

>>> cursor.curr()

3

>>> cursor.prev()

2

>>> cursor.prev()

1

>>> cursor.curr()

1

6. Write a class named Cursor that models an integer “cursor”. When created, the cursor object should have value 0. The class should support method curr() that returns the value of the cursor, method next() that increments the value of the cursor, and method prev() that decrements the value of the cursor. Metholds prev() and next() should also return the new value of the cursor. A ValueError exception should be raised if method prev() is called when the cursor value is 0.

>>> cursor = Cursor()

>>> cursor.curr()

0

>>> cursor.next()

1

>>> cursor.curr()

1

>>> cursor.prev()

0

>>> cursor.curr()

0

>>> cursor.prev()

Traceback (most recent call last):

File "<pyshell#169>", line 1, in <module>

cursor.prev()

File "/Users/me/ch8.py", line 54, in prev

raise ValueError('Illegal cursor operation')

ValueError: Illegal cursor operation

7. Write classes named Cursor and NegativeCursorError that model an integer “cursor” and an exception when an attempt is made to move the cursor out of bounds. When created, the cursor object should have value 0. The class should support method curr() that returns the value of the cursor, method next() that increments the value of the cursor, and method prev() that decrements the value of the cursor. Metholds prev() and next() should also return the new value of the cursor. A NegativeCursorError exception should be raised if method prev() is called when the cursor value is 0.

>>> cursor = Cursor()

>>> cursor.next()

1

>>> cursor.curr()

1

>>> cursor.prev()

0

>>> cursor.curr()

0

>>> cursor.prev()

Traceback (most recent call last):

File "<pyshell#180>", line 1, in <module>

cursor.prev()

File "/Users/me/ch8.py", line 72, in prev

raise NegativeCursorError()

NegativeCursorError

8. Write a class named Selection with a constructor that takes a list as an input. The class should have a method next() that returns the next item in the list, in a round-robin fashion.

>>> lst = ['Zoe', 'Yannick', 'Xena', 'Wendy']

>>> sel = Selection(lst)

>>> sel.next()

'Zoe'

>>> sel.next()

'Yannick'

>>> sel.next()

'Xena'

>>> sel.next()

'Wendy'

>>> sel.next()

'Zoe'

9. Write a class named Lottery that can be used to select lottery numbers. It should have a constructor that takes as input a positive integer n. The class should also have a method next() that returns a number selected uniformly at random and without replacement from the set of positive integers in the range from 1 to n.

>>> lotto = Lottery(60)

>>> lotto.next()

9

>>> lotto.next()

8

>>> lotto.next()

23

>>> lotto.next()

2

>>> lotto.next()

11

>>> lotto.next()

32

10. Write a class named Address that represents a postal address. It should have a constructor that takes as input a person’s first name, last name, street address, city, state, and zip code. The class should behave as shown below.

>>> dad = Address('John', 'Smith', '123 Main St', 'Anytown', 'ZX', '12345')

>>> dad

John Smith

123 Main St

Anytown, ZX 12345

11. The class Counter is defined as follows:

class Counter(object):

def \_\_init\_\_(self, i = 0):

self.i = i

def next(self):

res = self.i

self.i += 1

return res

It behaves as follows:

>>> i = Counter(1)

>>> i

<\_\_main\_\_.Counter object at 0x10475d898>

>>> j = Counter(1)

>>> i == j

False

Modify class Counter so it behaves as follows:

>>> i = Counter(1)

>>> i

1

>>> j = Counter(1)

>>> i == j

True

12. Write a class named myInt that behaves just like the built-in class int except for the addition overloaded operator.

>>> x = myInt(3)

>>> x - 1

2

>>> x \* 4

12

>>> x / 2

1.5

>>> x + 2

1

13. Write a class named myInt that behaves just like the built-in class int except for the random behavior of the equal operator shown below, which returns True or False with equal probability, regardless of the values being compared:

>>> x = myInt(3)

>>> x == 3

True

>>> x == 3

True

>>> x == 3

False

>>> x == 3

True

>>> x == 3

False

14. Write a class named Marsupial that can be used as shown below:  
  
>>> m = Marsupial()  
>>> m.put\_in\_pouch('doll')  
>>> m.put\_in\_pouch('firetruck')  
>>> m.put\_in\_pouch('kitten')  
>>> m.pouch\_contents()  
['doll', 'firetruck', 'kitten']

Now write a class named Kangaroo as a subclass of Marsupial that inherits all the attributes of Marsupial and also:

1. *extends* the Marsupial \_\_init\_\_ constructor to take, as input, the coordinates x and y of the Kangaroo object,
2. *supports* method jump that takes number values dx and dy as input and moves the kangaroo by dx units along the x-axis and by dy units along the y-axis, and
3. *overloads* the \_\_str\_\_ operator so it behaves as shown below.

>>> k = Kangaroo(0,0)  
>>> print(k)  
I am a Kangaroo located at coordinates (0,0)  
>>> k.put\_in\_pouch('doll')  
>>> k.put\_in\_pouch('firetruck')  
>>> k.put\_in\_pouch('kitten')  
>>> k.pouch\_contents()  
['doll', 'firetruck', 'kitten']  
>>> k.jump(1,0)  
>>> k.jump(1,0)  
>>> k.jump(1,0)  
>>> print(k)  
I am a Kangaroo located at coordinates (3,0)