Final Sample 1 Solutions

1. While Loop Simulation

Function Call	Output
`mystery(5, 0)`	`5`
`mystery(3, 2)`	`1 0 1`
`mystery(16, 5)`	`3 2 1 0 1`
`mystery(80, 9)`	`8 4 2 1 2 0 2`
`mystery(1600, 40)`	`40 19 2 9 0 4 0`

2. Inheritance Mystery

```
b
c 1
a 2  c 1

b
c 1
b 2  c 2

c
c 1
c 2

b
d 1  b 2  c 2
b 2  c 2
```

3. Collections Mystery

Function Call	Contents of Set Returned
`mystery(grid, 2, 2)`	`[6, 7]`
`mystery(grid, 0, 2)`	`[1, 2, 5, 8]`
`mystery(grid, 3, 3)`	`[1, 2, 3, 7, 9]`

4. List Programming

```
def is_unique(list):
    for i in range(1, len(list)):
        for j in range(i, len(list)):
            if (list[i - 1] == list[j]):
                return False
    return True
```

5. Collections Programming

```
def count_in_area_code(numbers, area_code):
    unique_numbers = set()
    for name, phone in numbers.items():
        if (phone[0:3] == area_code):
            unique_numbers.add(phone)
    return len(unique_numbers)
```

6. Programming

```
    def same_pattern(s1, s2):
        if (len(s1) != len(s2)):
            return False
        for i in range(0, len(s1)):
            for j in range(i + 1, len(s1)):
                if (s1[i] == s1[j] and s2[i] != s2[j]):
                    return False
                if (s2[i] == s2[j] and s1[i] != s1[j]):
                    return False
        return True
```

7. 2d Lists

```
def find_max(lis):
    max_sum = 0
    max_row = 0
    for i in range(0, len(lis)):
        cur_sum = 0
        cur_row = i
        for j in range(0, len(lis[i])):
            cur_sum += lis[i][j]
        if cur_sum > max_sum:
            max_sum = cur_sum
            max_row = cur_row
    return max_row
```

8. Critters

```
   class Ostrich(Critter):
       def __init__(self):
           super(Ostrich, self).__init__()
           self.__hiding = True
           self.__steps = 0
           self.__west = randint(0, 1) == 0

       def get_color(self):
           if (self.__hiding):
               return "cyan"
           else:
               return "white"

       def get_move(self):
           if (self.__steps == 10):
               self.__steps = 0   # Pick a new direction and re-set the steps counter
               self.__hiding = not self.__hiding
               self.__west = randint(0, 1) == 0

           self.__steps += 1
           if (self.__hiding):
               return DIRECTION_CENTER
           elif (self.__west):
               return DIRECTION_WEST
           else:
               return DIRECTION_EAST
```

9. Classes

```
   def compare(other):
       if (self.__month < other.__month or (self.__month == other.__month and
            self.__day < other.__day)):
           return -1
       elif (self.__month == other.__month and self.__day == other.__day):
           return 0
       else:
           return 1
```