class Course(object):
    """An instance represents an offering of a course at Cornell. There is a separate Course instance for each semester in which a course is offered. Each course also keeps track of the students who are enrolled."
    
    Instance variables:
    title (str) -- title of course
    credits [int] -- number of credits
    students [list of Student] -- list of students enrolled in this course
    ...

    def __init__(self, title, credits):
        """A new course with the given title and number of credits."
        Pre: title is a string (e.g., 'CS1110: Awesome Introduction to Python')
        credits is a positive integer
    
    self.title = title
    self.credits = credits
    self.students = []

    class Schedule(object):
        """Instances represent a student’s schedule for one semester."
        
        Instance variables:
        student (Student) -- the student whose schedule this is
        semester (str) -- the semester this schedule is for
        courses [list of Course] -- the Courses in this schedule
        ...

        def __init__(self, student, semester):
            """A new schedule for <student> in <semester>, which starts with no courses."
            Pre: <student> is a string.
        
        self.student = student
        self.semester = semester
        self.courses = []

    def total_credits(self):
        """Return: the total number of credits in this schedule."
        
        total = 0
        for course in self.courses:
            total += course.credits
        return total

    def overlaps(self, other_schedule):
        """Return: True if this schedule contains any course with the same title as a course contained in <other_schedule>."
        Pre: <other_schedule> is a Schedule.
        """

        for course in self.courses:
            if other_schedule.contains_course(course):
                return True
        return False

    def contains_course(self, query_course):
        """Return: True if this schedule contains a course with the same title as <query_course>."
        
        for course in self.courses:
            if course.title == query_course.title:
                return True
        return False

class Student(object):
    """Instances represent students at Cornell. For each student, we keep track of their schedules for each semester they’ve been at Cornell."
    
    Instance variables:
    name (str) --- Name of student
    schedules [list of Schedule] -- the student’s schedules from all semesters, in reverse chronological order. The Schedule for the current semester is at position 0 in this list.
    ...

    def __init__(self, name):
        """A new student named <name>, who starts with no schedules."
        Pre: <name> is a string.
    
    self.name = name
    self.schedules = []

    def start_semester(self, semester):
        """Set up for a new semester by adding an empty Schedule at the head of the schedules list."
        Pre: <semester> is a string, such as '2014sp'
    
    self.schedules.insert(0, Schedule(self, semester))

    def add_course(self, course):
        """Add a course for the current semester. This means the course is added to the student’s current schedule, and the student is added to the enrollment of the course."
        Pre: <course> is a Course, the student has a current schedule, and <course> is not already on the current semester’s schedule.
    
        # TODO: implement this method

    def validate(self, credit_limit):
        """Return: True if the student’s schedule for the current semester is valid, which means that (a) the total number of credits in the current semester is not over <credit_limit> credits from prior semesters don’t matter) (b) the student is not taking any courses in the current semester that they already took in a previous semester. Course titles are used to determine when a course is repeated; see Schedule.overlaps.
        Pre: credit_limit is an integer, and the student has a current schedule.
    
        # TODO: implement this method

    def test_enrollment(self):
        """Test the enrollment system, making sure particularly that validation of schedules works properly and that students get enrolled in the courses that go on their schedules."
        
        # Four courses, offered in each of two semesters
        c1_s14 = Course('CS1110: Awesome Python', 4)
        c2_s14 = Course('CS2110: Jolly Java', 4)
        c3_s14 = Course('CS4740: Natural Language Processing', 4)
        c4_s14 = Course('CS4620: Computer Graphics', 3)
        c1_f14 = Course('CS1110: Awesome Python', 4)
        c2_f14 = Course('CS2110: Jolly Java', 4)
        c3_f14 = Course('CS4740: Natural Language Processing', 4)
        c4_f14 = Course('CS4620: Computer Graphics', 3)
# A student whose course enrollment validates OK
ljl = Student('Lillian Lee')
ljl.start_semester('Spring 2014')
ljl.add_course(c1_s14)
ljl.start_semester('Fall 2014')
ljl.add_course(c2_f14)
assert ljl.schedules[1].contains_course(c1_s14)
assert not ljl.schedules[1].contains_course(c2_f14)
assert not ljl.schedules[1].overlaps(ljl.schedules[1])
assert ljl.schedules[0].total_credits() == 4
assert ljl.validate(5)

# A student who is trying to re-take a course
srm = Student('Steve Marschner')
srm.start_semester('Spring 2014')
srm.add_course(c1_s14)
srm.start_semester('Fall 2014')
srm.add_course(c1_f14)
srm.add_course(c2_f14)
assert srm.schedules[1].contains_course(srm.schedules[0].courses[0])
assert srm.schedules[1].overlaps(srm.schedules[0])
assert not srm.validate(5)

# A student who is trying to take too many credits
mcp = Student('Mary Pisaniello')
mcp.start_semester('Fall 2014')
mcp.add_course(c1_f14)
mcp.add_course(c2_f14)
mcp.add_course(c3_f14)
mcp.add_course(c4_f14)
assert mcp.schedules[0].total_credits() == 15
assert not mcp.validate(14)

# Check that enrollments came out OK
assert set(c1_s14.students) == set([ljl, srm])
assert set(c2_f14.students) == set([ljl, mcp])

if __name__ == '__main__':
test_enrollment()