```python
class Course():
    """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] -- students enrolled in course"""

    def __init__(self, title, credits):
        """A new course with the given title and number of credits.
        The course starts out with no students enrolled.
        Pre: title is a string (e.g., 'CS1110: Awesome Python')
             credits is a positive integer"""
        self.title = title
        self.credits = credits
        self.students = []


class Schedule():
    """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):
        """Schedule for <student> in <semester>. Starts with no courses.
        """
        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():
    """Instances represent students at Cornell.  For each student, we
    track 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.  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 '2018sp'"""
        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 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 current semester is not over
               <credit_limit> (credits from prior semesters don't matter)
           (b) student is not taking any courses in current semester that
               they already took in a previous semester. Course titles
               determine when a course is repeated; see Schedule.overlaps.
        Pre: credit_limit [integer] ; student has a current schedule."""
        # TODO: implement this method
        # Take the time to read through all the methods in Schedule:
        # using them makes this method much shorter to implement.
```

```python

def test_enrollment():
    """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_s18 = Course('CS1110: Awesome Python', 4)
    c2_s18 = Course('CS2110: Jolly Java', 4)
    c3_s18 = Course('CS4740: Natural Language Processing', 4)
    c4_s18 = Course('CS4620: Computer Graphics', 3)
    c1_f18 = Course('CS1110: Awesome Python', 4)
    c2_f18 = Course('CS2110: Jolly Java', 4)
    c3_f18 = Course('CS4740: Natural Language Processing', 4)
    c4_f18 = Course('CS4620: Computer Graphics', 3)

    # A student whose course enrollment validates OK
    s1 = Student('Lillian Lee')
    s1.start_semester('Spring 2018')
    s1.add_course(c1_s18)
    s1.start_semester('Fall 2018')
    s1.add_course(c2_f18)
    assert s1.schedules[1].contains_course(c1_s18)
    assert not s1.schedules[1].contains_course(c2_f18)
    assert not s1.schedules[0].overlaps(s1.schedules[1])
    assert s1.schedules[0].total_credits() == 4
    assert s1.validate(5)

    # A student who is trying to re-take a course
    s2 = Student('Steve Marschner')
    s2.start_semester('Spring 2018')
    s2.add_course(c1_s18)
    s2.start_semester('Fall 2018')
    s2.add_course(c1_f18)
    assert s2.schedules[1].contains_course(s2.schedules[0].courses[0])
    assert s2.schedules[1].overlaps(s2.schedules[0])
    assert not s2.validate(5)

    # A student who is trying to take too many credits
    s3 = Student('Mary Pisaniello')
    s3.start_semester('Fall 2018')
    s3.add_course(c1_f18)
    s3.add_course(c2_f18)
    s3.add_course(c3_f18)
    s3.add_course(c4_f18)
    assert s3.schedules[0].total_credits() == 15
    assert not s3.validate(18)

    # Check that s1 & s2 are enrolled in c1_s18
    assert set(c1_s18.students) == set([s1, s2])
    # Check that s1 & s3 are enrolled in c2_f18
    assert set(c2_f18.students) == set([s1, s3])


if __name__ == '__main__':
    test_enrollment()

```