TastySoup is a soup and salad restaurant located between UCLA and USC campuses. Given that the restaurant is located near UCLA and USC, the restaurant wants to design a frequent dining program to attract student customers and encourage learning. The rules for the frequent dining program are as follows:

- Customers purchase a meal that is 15.00 dollars or more will get ten points credited to their account. They will get an extra 5 points for every 15.00 dollars (for example, $15 gets 10 points, $30 gets 15 points, $45 gets 20 points, etc.). No points will be given if the meal is less than $15.
- UCLA students will get five bonus points for each meal
- USC students will get three bonus points for each meal
- Students from other schools will get zero bonus point
- Students who can provide evidence that their CS 101 midterm grade is above 80 will get three extra points
- If the student cannot remember their password, then their frequent point balance will be rest to zero (also, no points will be given for the current meal).

Your Task:

1. Create two class objects: Account and Order (Order is a subclass of Account)
2. Account has the attributes of cust_num (4 digit integer customer number – for example, 1000, 1001, 1002) and u_name (university name – for example, UCLA, USC, Others)
3. In addition to the cust_num and u_name attributes inherited from Account, Order has the attributes of midterm (integer number - for example, 56, 78, 98, 100), password (a 4 digit integer number. Enter 0 if student forgets their password), meal_cost (how much the customer is paying for the meal), and balance (the total frequent dining points customers have in their accounts). Order only needs to initialize the additional attributes since it let Account handles the attributes associated with the Account class. midterm and password are private information; therefore, those two attributes need to be private member attributes.
4. Create these five methods in the Order class:
    a. UpdateBalance(self)
        i. To update the frequent point balance based on the rules stated above
    b. getBalance(self)
        i. returns balance to the caller
    c. getMidterm(self)
        i. returns midterm to the caller
    d. getPassword(self)
        i. returns password to the caller
    e. updateMidterm(self, midterm)
        i. Sets midterm to the new value

Here is an example driver code to test your function. You should test your code with more test data:

```
acct1 = Account(1000, 'UCLA')

order1 = Order(1000, 'UCLA', 89, 9878, 16.95, 0)

order1.UpdateBalance()
```

order1.UpdateMidterm(100)

Save your Python file as tastysoup.py