2、The `Meal` class and its subclass `DeluxeMeal` are used to represent meals at a restaurant.

All `Meal` objects have the following attributes and methods.

A `String` variable representing the name of the entree included in the meal
A `double` variable representing the cost, in dollars, of the meal
A `toString` method that indicates information about the meal

The following table shows the intended behavior of the `Meal` class.

Statement	Result
`Meal burger = new Meal("hamburger", 7.99);`	A new `Meal` object is created to represent a hamburger that costs $7.99.
`burger.toString();`	The string `"hamburger meal, $7.99"` is returned.

(a) Write the complete `Meal` class. Your implementation must meet all specifications and conform to the behavior shown in the table.
题目给的代码是java的，需要改为python，比如把new去掉

A deluxe meal, represented by a `DeluxeMeal` object, includes a side dish and a drink for an additional cost of $3. The `DeluxeMeal` class is a subclass of `Meal`.
The `DeluxeMeal` class contains two additional attributes not found in `Meal`:

A `String` variable representing the name of the side dish included in the meal
A `String` variable representing the name of the drink included in the meal

The following table shows the intended behavior of the `DeluxeMeal` class.

Statement	Result
`DeluxeMeal burritoCombo = new DeluxeMeal("burrito", "chips", "lemonade", 7.49);`	A new `DeluxeMeal` object is created to represent a deluxe meal including a burrito (entree), chips (side dish), and lemonade (drink). The cost of the burrito alone is $7.49, so the cost of the meal is set to $10.49.
`burritoCombo.toString();`	The string `"deluxe burrito meal, $10.49"` is returned.

(b) Write the complete `DeluxeMeal` class. Your implementation must meet all specifications and conform to the behavior shown in the table.

2. The `Book` class is used to store information about a book. A partial `Book` class definition is shown.

```
class Book:

    /** The title of the book */
    #private String title

    /** The price of the book */
    #private double price

    /** Creates a new Book with given title and price */
    def __init__(self, bookTitle, bookPrice):

       /* implementation not shown */

    /** Returns the title of the book */
    def getTitle(self):
     return self.__title

    /** Returns a string containing the title and price of the Book */
    def getBookInfo(self):
      return self.__title + "-" + self.__price

   // There may be instance variables, constructors, and methods that are not shown.
```

You will write a class `Textbook`, which is a subclass of `Book`.

A `Textbook` has an edition number, which is a positive integer used to identify different versions of the book. The `getBookInfo` method, when called on a `Textbook`, returns a string that also includes the edition information, as shown in the example.

Information about the book title and price must be maintained in the `Book` class. Information about the edition must be maintained in the `Textbook` class.

The `Textbook` class contains an additional method, `canSubstituteFor`, which returns `true` if a `Textbook` is a valid substitute for another `Textbook` and returns `false` otherwise. The current `Textbook` is a valid substitute for the `Textbook` referenced by the parameter of the `canSubstituteFor` method if the two `Textbook` objects have the same title and if the edition of the current `Textbook` is greater than or equal to the edition of the parameter.

GO ON TO THE NEXT PAGE.

The following table contains a sample code execution sequence and the corresponding results. The code execution sequence appears in a class other than `Book` or `Textbook`.

Statement	Value Returned (blank if no value)	Class Specification
`bio2015 = Textbook("Biology", 49.75, 2);`		`bio2015` is a `Textbook` with a title of `"Biology"`, a price of `49.75`, and an edition of `2`.
`bio2019 = Textbook("Biology", 39.75, 3);`		`bio2019` is a `Textbook` with a title of `"Biology"`, a price of `39.75`, and an edition of `3`.
`bio2019.getEdition();`	`3`	The edition is returned.
`bio2019.getBookInfo();`	`"Biology-39.75-3"`	The formatted string containing the title, price, and edition of `bio2019` is returned.
`bio2019.canSubstituteFor(bio2015);`	`true`	`bio2019` is a valid substitute for `bio2015`, since their titles are the same and the edition of `bio2019` is greater than or equal to the edition of `bio2015`.
`bio2015.canSubstituteFor(bio2019);`	`false`	`bio2015` is not a valid substitute for `bio2019`, since the edition of `bio2015` is less than the edition of `bio2019`.
`math = Textbook("Calculus", 45.25, 1);`		`math` is a `Textbook` with a title of `"Calculus"`, a price of `45.25`, and an edition of `1`.
`bio2015.canSubstituteFor(math);`	`false`	`bio2015` is not a valid substitute for `math`, since the title of `bio2015` is not the same as the title of `math`.

Write the complete `Textbook` class. Your implementation must meet all specifications and conform to the examples shown in the preceding table.

Begin your response at the top of a new page in the Free Response booklet and fill in the appropriate circle indicating the question number.
If there are multiple parts to this question, write the part letter with your response.

GO ON TO THE NEXT PAGE.

附加题，自己选择是否做，预计掉10根头发哈哈，是一道竞赛题

Sample Input	Sample Output
15 8 2	9
25 2 1111011	105
20 12 9AB	14
10 16 ABCDEF	10
1000 2 1	4938

问题：给定 3 个正整数 *n*、*b* 和 *s*，生成以给定进制的数字 *s* 开始的接下来 *n* 个 *b* 进制数字。我们确保进制介于 2 和 16 之间（含 2 和 16）并且 *s* 是一个 *b* 进制的有效数字。找出生成数字的各个数位上的所有数字并打印输出出现频率最高的数字对应出现总次数的十进制数。

示例：如果 *n* = 15, *b* = 8 且 *s* = 2，生成的数字是 2、 3、 4、 5、 6、 7、 10、11、12、13、14、15、 16、 17、20。出现频率最高的数字是 1，一共出现了 9 次，所以输出的值是 9。

输入：将会有三个整数，分别表示生成值的数量，要使用的进制（介于 2 和 16 之间，包含 2 和 16），给定进制的起始值（不超过16 位）。我们保证 *s* 是一个 *b* 进制下的有效数字。

输出：对于每组3 个输入值，求生成的数列中每个数字出现的次数，输出出现频率最高的数字出现的次数。