CSCI-1200 Data Structures — Spring 2013
Homework 4 — Grocery Lists

In this assignment you will write a program to manage the contents of a kitchen as various recipes are prepared. Your program will handle several different operations: adding a recipe, buying ingredients for the kitchen, printing out a recipe, making a recipe (which removes ingredients from the kitchen), printing out the current contents of the kitchen, and suggesting recipes that can be made from the current contents of the kitchen. *Please carefully read the entire assignment before beginning your implementation.*

The input for the program will come from a file and the output will also go to a file. These file names are specified by command-line arguments. Here's an example of how your program will be called:

```
grocery_list.exe requests.txt results.txt
```

The form of the input is relatively simple. Each request begins with a single character, indicating one of the following requests, described below. You may assume the input file strictly follows this format (i.e., you don't need to worry about format error-checking).

```
r salad
2 tomatoes
1 lettuce
0
a
3 lettuce
4 tomatoes
0
p salad
m salad
k
```

- The first request, indicated by the letter '`r`', says to add a particular recipe to the list of recipes. The ingredients follow, listed one per line with the quantity and name. The ingredient list ends with a '`0`' (zero) on a line by itself. You should verify that there is not already a recipe with this name in the list of recipes. The output for this request is:

  ```
  Recipe for salad added
  ```

  Or, if a recipe with that name was previously entered into in the system:

  ```
  Recipe for salad already exists
  ```

- The next request, indicated by the letter '`a`', says that various ingredients should be added to the kitchen supplies. Again, the ingredients follow, listed one per line with the quantity & name, and the ingredient list ends with a '`0`' (zero). When you add an ingredient to the kitchen you need to first check if the ingredient is already present. If it is, simply increase the quantity of that item. The output for this request indicates the number of distinct ingredients (not the quantity) that were added:

  ```
  2 ingredients added to kitchen
  ```

- The '`p`' command, asks that the recipe for `salad` be output. The ingredients should be listed in alphabetical order with their quantities. For example:

```
To make salad, mix together:
  1 unit of lettuce
  2 units of tomatoes
```

  Note the user friendly output text distinguishes between singular and plural quantities. If the program does not have the requested recipe, the following message should be output:

```
No recipe for salad
```

- The next request, indicated by the letter '`m`', asks the program to attempt to make a particular recipe. First the program will check the kitchen to see if the necessary ingredients are available. If all of the ingredients are available in sufficient quantities, the kitchen will be edited to remove the appropriate amounts of each ingredient and this message is output:

```
Made salad
```

  If the ingredients are not available, the output message will list the insufficient ingredients (sorted alphabetically), with the quantities that are missing. In this example, one salad can be made, but if a second salad is requested, the following message will be output:

```
Cannot make salad, need to buy:
  1 unit of tomatoes
```

  If the program does not have the requested recipe, the following message should be output:

```
Don't know how to make salad
```

- The fifth request, indicated by the letter '`k`', asks for the current contents of the kitchen to be output, sorted by quantity first, and then alphabetically for items with equal quantity. In this example:

```
In the kitchen:
  1 unit of tomatoes
  3 units of lettuce
```

  If a particular ingredient has been "used up", it should be removed from the list of ingredients.

- The final required request for the main homework, indicated by the letter '`s`', asks the system to suggest recipes from its collection of known recipes that can individually be made from the current ingredients of the kitchen. These recipes should be listed sorted alphabetically by recipe name. For a well-stocked kitchen, the output might look like this:

```
Recipes that can be prepared:
  cake
  cookies
  quiche
  salad
```

  This command does not prepare any of the recipes, and thus does not modify the kitchen ingredients.

- **For extra credit**, your solution can also handle the 'd' request that outputs a suggestion of the combination of different recipes that can all be made for dinner that would use up the maximum total units of ingredients.

```
Menu suggestion for dinner:
  cake, quiche, and salad
```

  Note that the dinner menu may indeed include fewer items than the output from the suggestions list above (request 's'). This may happen if, for example, we do not have enough sugar to make both cake and cookies. In this case, we would suggest the dinner with cake, quiche, and salad over making cookies, quiche, and salad if the former requires more total units of ingredients. The dinner menu list is output in alphabetical order by recipe, and neatly presented in a the human readable format, as shown above.

Sample input and output files are posted on the course web site. Please follow these examples exactly to aid in the automatic grading of your work. You can use the UNIX `diff` command to compare your output to the sample output files.

Order Notation

You should implement the functionality above with efficiency in mind. In your README.txt file, use order notation to analyze the computation required to process each of the different requests. In your analysis use:

i = # of different ingredients in the kitchen
u = maximum units of a single ingredient in the kitchen
r = # of different recipes
k = maximum # of different ingredients in a single recipe
v = maximum units of single ingredient in a single recipe
a = maximum # of different ingredients added with a single 'a' command
w = maximum units of a single ingredient added with a single 'a' command

Note: You may not need to use all of these variables in your answers.

Additional Requirements, Hints and Suggestions

You may not use vectors or arrays for this assignment. Use the standard library (STL) lists and iterators instead. You may not use maps, or sets, or things we haven't discussed in lecture yet.

You must write at least one new class. We have provided a partial implementation of the main program to get you started. There are member function calls to our versions of the `Recipe` and `Kitchen` classes, so you can deduce how some of the member functions in our solution work. You may use none, a little, or all of this, as you choose, but we strongly urge you to examine it carefully.

Submission

Do all of your work in a new folder named `hw4` inside of your Data Structures homeworks directory. Use good coding style when you design and implement your program. Be sure to make up new test cases and don't forget to comment your code! Please use the provided template `README.txt` file for any notes you want the grader to read. **You must do this assignment on your own, as described in the "Academic Integrity for Homework" handout. If you did discuss the problem or error messages, etc. with anyone, please list their names in your `README.txt` file.** When you are finished please zip up your files exactly as instructed for the previous assignments and submit it through the course webpage.