APS 105 — Computer Fundamentals

Lab 9: Search and Link
Winter 2020

In this lab you will take a program skeleton and complete various parts to make the solution. The final solution will be a set of cooking recipes from which you can get information as expected.

Recipe List

You are going to complete a recipe utility that stores recipe ingredients, adds new recipes, prints out the recipe list and prints out a shopping list for selected recipes. For this exercise the recipe will consist only of the ingredient list so will **not** include preparation and cooking directions.

The command menu for this utility is as follows:

```
Enter a command by letter
1. List all Recipes
2. Print a Recipe
3. List All Recipes with a given ingredient
3. List All Ingredients in alphabetical order
4. Exit the program
Give input:
```

You will be responsible for implementing these commands.

A populated list will be provided to you as ”rawRecipes.h” that you should include in your program:

```
    #include "rawRecipes.h"
```

You should also include the .h file in the same directory as your program.

In this .h file a recipe name is preceded by a 0 and this is followed by a varying number of ingredient names which are preceded by a 1. This is a dump of the beginning lines of the file:

```
char *rawRecipes[]={
"0Broccoli Coleslaw",
"1olive oil",
"1white vinegar",
"1white sugar",
"1package chicken flavored ramen noodles",
"1broccoli",
"1carrots",
"1green onions",
"1sunflower seeds",
"0Creamy Broccoli Salad",
"1broccoli",
"1red onion",
...
```

Note that the first recipe is ”Broccoli Coleslaw” and the next is ”Creamy Broccoli Salad”. Both of these include the ingredient ”broccoli”. You will find this is a common theme in the recipes!

Programming requirements and notes:

- You should only scan the .h file information once (as if it came from a disk drive file)
- You must use a ”struct” in your program
- There will be a small portion marks allocated to how concisely you store the data, for example only storing the ingredient string ”broccoli” once.

Programming hints:

- Look at what you have to do with the list. You might want to create data structures that are aligned with the manipulations you need to perform.
- Work with rawRecipesDevt.h (provided), which is a truncated version of rawRecipes.h during your development. Since it is smaller, you can dump the results and check out your ”end of list” code without going through a large amount of input.
- Write a routine for debugging/development purposes only that will dump out your data structure in a way that allows you to easily tell what is working and what is not working. Remove the calls to this routine for code checks and submission at the end.

IMPORTANT: When you submit your code for automarking, make sure the included file is rawRecipes.h, spelt EXACTLY like that, including uppercase and lowercase letters. Otherwise your code may not compile in the automarker and you will get a zero (0), even though it might work on your laptop or in the lab.

Required Output Format for Auto-Marking

As before, the automarker will be looking for exactly a certain output format. Note that the menu itself begins with 2 ”newline” characters. Also note that these outputs in some cases have middle contents replaced by a ”...” to save space. Refer to rawRecipes.h to see where the information comes from that is printed.

Menu item 1. List all Recipes

```


Enter a command by number
1. List all Recipes
2. Print a Recipe
3. List All Recipes with a given ingredient
4. List All Ingredients in alphabetical order
5. Exit the program
Give input: 1
Recipe #1: Broccoli Coleslaw
Recipe #2: Creamy Broccoli Salad
Recipe #3: Minnesota Broccoli Salad
...
Recipe #23: Strawberry Broco-Flower Salad
Recipe #24: Broccoli and Tortellini Salad
```

Menu item 2. Print a Recipe

```


Enter a command by number
1. List all Recipes
2. Print a Recipe
3. List All Recipes with a given ingredient
4. List All Ingredients in alphabetical order
5. Exit the program
Give input: 2

Give the number of the recipe (1-24):6
Garlic Broccoli Salad
broccoli
olive oil
pine nuts
rice wine vinegar
vegetable oil
cloves garlic
cayenne pepper
raisins
```

Menu item 3. List All Recipes with a given ingredient

```
Enter a command by number
1. List all Recipes
2. Print a Recipe
3. List All Recipes with a given ingredient
4. List All Ingredients in alphabetical order
5. Exit the program
Give input: 3

Give the name of the ingredient:olive oil
Broccoli Coleslaw
Garlic Broccoli Salad
```

Menu item 4. List All Ingredients in alphabetical order

```
Enter a command by number
1. List all Recipes
2. Print a Recipe
3. List All Recipes with a given ingredient
4. List All Ingredients in alphabetical order
5. Exit the program
Give input: 4
ALL INGREDIENTS
0: Cheddar cheese
1: Dijon mustard
2: Italian dressing
3: Italian-style salad dressing
4: agave nectar
5: almonds
6: apple
7: apple cider vinegar
8: artichoke hearts
...
91: tomatoes
92: vegetable oil
93: white sugar
94: white vinegar
95: white wine vinegar
96: yogurt
```

Submitting Your Program for Auto-Marking

The total of **6 marks** on this lab:

1. **By an auto-marking program for 6 marks out of 6**. You must submit all of your program files through the ECF computers for marking. Long before you submit your program for marking, you should run the **exercise** program that compiles and runs your program and gives it sample inputs, and checks that the outputs are correct. Similar to previous labs you should run the following command:

   ```
   /share/copy/aps105s/lab4/exercise
   ```

   within the directory that contains your program.

   This program will look for the file `Lab9.c` in your directory, compile it, and run it on a some of the test cases that will be used to mark your program automatically later. If there is anything wrong, the **exercise** program will report this to you, so read its output carefully, and fix the errors that it reports.

2. Once you have determined that your program is as correct as you can make it, then you must submit your program for auto-marking. This must be done by the end of your lab period as that is the due time. To do so, go into the directory containing your program and type the following command:

   ```
   /share/copy/aps105s/lab9/submit
   ```

   This command will re-run the exercise program to check that everything looks fine. If it finds a problem, it will ask you if you are sure that you want to submit. Note that you may submit your work as many times as you want prior to the deadline; only the most recent submission is marked.

   The **exercise** program (and the **marker** program that you will run after the final deadline) will be looking for the exact letters as described in the output in this handout, including the capitalization. When you test your program using the exercise program, you will see that it is expecting the output to be exactly this, so you will have to use it to see if you have this output correct.

   Important Note: You must submit your lab by the end of your assigned lab period. Late submissions will not be accepted, and you will receive a grade of zero.

   You can also check to see if what you think you have submitted is actually there, for peace of mind, using the following command:

```
/share/copy/aps105s/lab9/viewsubmitted
```

This command will download into the directory you run it in, a copy of all of the files that have been submitted. If you already have files of that same name in your directory, these files will be renamed with a number added to the end of the filename.

After the Final Deadline — Obtaining Automark

Briefly after all lab sections have finished (after the end of the week), you will be able to run the automarker to determine the automarked fraction of your grade on the code you have submitted. To do so, run the following command:

```
/share/copy/aps105s/lab9/marker
```

This command will compile and run your code, and test it with all of the test cases used to determine the automark grade. You will be able to see those test cases output and what went right or wrong.