AU25 CSE 2421 PROJECT 3

Assigned: Friday, September 19th
Early Due Date: Monday, September 29th at noon (10% bonus)
Due: Tuesday, September 30th, at 11:30 p.m.
Any project submitted up to 24 hours late will be graded and points earned multiplied by 0.75.
Any project submitted more than exactly 24 hours from the due date will not be graded and scored as a 0.

Life is really simple, but we insist on making it complicated. – Confucius

(And this project is a lot like life.)

Objectives:

· Pointers
· Multiple Levels of indirection
· Dynamic memory allocation/deallocation
· Functions
· Arrays (dynamically allocated)
· Character Strings

REMINDERS and GRADING CRITERIA:

- This is an individual project.
- Waiting until the day this project is due to start working on it, would be an unbelievably bad idea.
- Effort has been made to ensure that this project description is complete and consistent. That does not mean that you will not find errors or inconsistency. If you do, please ask for clarification.
- **Every project requires a Readme file** (for this project, it should be called **Project3Readme – use this name.** This file should include the following:

    · Required Header:
    BY SUBMITTING THIS FILE TO CARMEN, I CERTIFY THAT I HAVE STRICTLY ADHERED TO THE TENURES OF THE OHIO STATE UNIVERSITY'S ACADEMIC INTEGRITY POLICY WITH RESPECT TO THIS ASSIGNMENT.
    THIS IS THE README FILE FOR PROJECT 3.

    · Your name
    · Total amount of time (effort) it took for you to complete the project
    · Short description of any concerns, interesting problems or discoveries encountered, or comments in general about the contents of the project
    · Describe, with 4-5 sentences, how you used gdb to find a bug in your program while debugging it. Or, if you had no bugs in your program, how you used gdb to verify that your program was working correctly. **Include how you set breakpoints, variables you printed out, what values they had, what you found that enabled you to fix the bug.**

- You should aim to always hand an assignment in on time or early. If you are late (even by a minute or less), you will receive 75% of your earned points for the designated grade if the assignment is submitted by 11:30 pm the following day, based on the due date given above. If you are more than 24 hours late, you will receive a zero for the assignment and your assignment will not be graded at all.

- Any project submitted that does not compile – without errors or warnings - and run **WILL RECEIVE AN AUTOMATIC GRADE OF ZERO**. No exceptions will be made for this rule - to achieve even a single point on a project, your code must minimally build (compile to an executable without errors or warnings) on coelinux and execute on coelinux without crashing or hanging.

Since a Makefile is required for this project, you must create the appropriate compile statements to create a project3main.o, get_ingredients.o, get_item.o, get_thispizza.o, save_info.o, and free_dmem.o files, which would then create a project3 executable. Graders will be downloading your project3.zip file from Carmen, unzipping it, and then executing **make** from a linux command line prompt. Your program must compile – without errors or warnings – via commands within the Makefile. The gcc options your program must be compiled with are: **-std=c99 -pedantic -Wimplicit-function-declaration -Wreturn-type -Wformat**; you may also want to include the **-g** options so that debug information is generated or the -c option so that compilation can stop after creating a .o file. **Given valid input, your program must also run without having a seg fault, other abnormal termination or hanging.**

- You are welcome to do more than what is required by the assignment as long as it is clear what you are doing and it does not interfere with the mandatory requirements.

PROJECT DESCRIPTION

PIZZA SHOP INGREDIENT LIST WITH A PIZZA ORDER (100%)

Mandatory filenames:	**project3main.c**
	get_ingredients.c
	get_item.c
	get_thispizza.c
	save_info.c
	free_dmem.c
	project3.h
Mandatory executable name:	**project3**

You must adhere to the 10-line limit. You may create other functions you find useful (some may help you adhere to the 10-line limit). If you do, put each of them in a separate .c file and add a prototype for each of the new functions in project3.h.

PROBLEM:
This is part of a program being developed for the local pizza shop. This pizza shop only uses fresh ingredients so the ingredients they use on their pizzas changes daily. The first part of the system is used by an employee who enters what fresh ingredients they have on hand at the beginning of each day. You will not know how many fresh ingredients are available each day until the employee keys it into the program. Once the employee tells the program how many fresh ingredients are available, the employee will then enter each fresh ingredient on a separate line. This part of the program would be accomplished by calling the get_ingredients() function

and any functions get_ingredients() chooses to call.

Once the ingredients have been entered, the second – COMPLETELY INDEPENDENT – part of the program runs. The second half of the program is used by someone ordering a pizza. The program will tell the user what fresh ingredients are available today with a number next to each ingredient. Then, the user will specify how many ingredients they want on their pizza, then ask the user to specify each of the ingredients by number. You may want to review Slide Deck 15, slides 31 through 38 to help you visualize how the data should be stored. The array **titles** in the slides would be similar to what I expect you to implement for the ingredient list. The **favorites** array in the slides would be similar to what I expect you to implement for the customer's specific pizza order. **You must use pointers within this project such that you have one array of char * that contains the pizza ingredients and an array of char ** for the specific pizza order.** The third item this program does is ask the user whether s/he want to store a copy of their information to a file on disk. If so, you program will ask for a file name and store the information within the file in the format shown in the example below. This part of the program would be implemented with get_thispizza(), save_info(), and free_dmem()

1. <u>Since this is only part of the program, for testing purposes (until we design/write more of the software for the pizza shop) we are going to combine the part used by the employee to input today's fresh ingredient list with the customer order part.</u> We'll set up the ingredients array, then create an ingredient list for the customer's specific pizza based on the ingredient list.

2. First, you must prompt the user to enter the number of fresh ingredients they plan to enter. The user will enter an integer greater than or equal to 1. (NOTE: Make a point to test your program to ensure entering just 1 ingredient works.)

3. You must dynamically allocate memory for an array that holds a character array pointer to each fresh ingredient string, then dynamically allocate enough memory to hold each individual ingredient string as each ingredient is read in. You can assume that there will be no ingredient with more than 60 characters. Remember that all character strings in C are null terminated and that you must allocate space in your string for that null character, in addition to the length of the string you wish to have. Also realize that you won't know how many of the 60 characters the ingredients will fill until after you read the ingredient string. That means that you should allocate the maximum amount of memory for each string.

4. You must prompt the user to enter each ingredient on a separate line. You must assume that each ingredient could contain more than one word and will be separated by a newline character from the next ingredient. You can assume that the user will enter the input in this format, so you do not need to check to make sure that the format of the input meets this description, and you do not need to reject input which is not properly formatted. You can also assume that the user's input is correct. If you do not completely understand this description jump to the bottom of this file and check out the example data.

5. You can assume that there will be no duplicate ingredients.

6. After reading in all the ingredients, your program must print all the ingredients back out with a number next to each one.

7. Now that we have an ingredient list, we'll move to the second part of the program where a customer will use the ingredient list to order a pizza. you must prompt the user to tell you whether they wish to order a small, medium or large, pizza.

8. Second, you must ask the user to pick how many ingredients s/he wishes to put on their pizza. This list will consist of a subset (up through all ingredients on the available ingredient list) of the fresh ingredients list.

9. Once you have this number, you will have to dynamically allocate enough space for an array of 8-byte addresses for the array for this pizza order.

10. The user will then specify by ingredient number which ingredients should be included on their pizza.

11. Your program must then print out the ingredients they want on their pizza along with a price for each item. Base price for a small pizza is $5.00, a medium pizza is $7.50, and a large pizza is $10.00. Each ingredient added to the pizza is $1.00 on a small pizza, $1.25 on a medium pizza, and $1.50 on a large pizza. In addition, your program must print out a price to the total pizza.

12. Next, your program must ask whether the user wishes to store in an ASCII file the information they have input. They indicate yes/no with 1 or 2.

13. If the user wishes to save the data, you must open a file and store the information in the file based on the format shown below, close the file, and then confirm to the user the data has been saved.

14. Prior to exiting the program, you must free() all dynamically allocated memory. You can determine whether you have managed to accomplish this with the valgrind tool.

15. GUIDANCE ONLY: If your two array pointers are declared in main() thus:

```
char **ingredients;       /* this is the pointer for the start of the ingredients[] array */
char ***thispizza;        /* this is the pointer for the start of  this pizza order array */
```

    and the following two lines are executed prior to you freeing all dynamically allocated memory (i.e calling free_dmem()) without causing a segmentation fault, you can be sure that your arrays are constructed correctly:

```
printf("First available ingredient is %s\n", *ingredients);
printf("First ingredient on this pizza is %s\n", **thispizza);
```

The graders will be inserting lines like this into your program to verify correct array construction so you might want to give it a try while you test.

REQUIREMENTS AND CONSTRAINTS:
· **ALL source code files (.c files and .h files) submitted to Carmen as a part of this program must include the following at the top of the file:**

```
/* BY SUBMITTING THIS FILE TO CARMEN, I CERTIFY THAT I HAVE STRICTLY ADHERED TO THE
TENURES OF THE OHIO STATE UNIVERSITY'S ACADEMIC INTEGRITY POLICY WITH RESPECT TO
THIS ASSIGNMENT.
*/
```

If you choose not to put the above comment in your file, you will receive no points for the project.

– You must comment your code

– You must adhere to items 1-14 above.

– You must create and submit a Makefile that will be used to create your .zip file and your executable, **project3**. This Makefile must define at least each of the following targets: **all**, **project 3**, **project3.zip**, **project3main.o**, **get_ingredients.o, get_item.o, get_thispizza.o, save_info.o, free_dmem.o, and clean**. **Creating this file at the beginning of your development process and using it as you work, will allow you to test your Makefile for proper operation.**

– You must separate your code into different functions so that your main() is uncluttered and easy to understand. Pointers to your two arrays must be declared in your main() program. You must have a different function that is called from main() that performs each of the following tasks:

1. populates the fresh ingredients array in a file called **get_ingredients.c**

   A. get_ingredients() must call a function called get_item() that reads a single fresh ingredient from input. get_item() must reside in a file called **get_item.c**

2. populates the thispizza array in a file called **get_thispizza.c**

3. saves data to a file in a file called **save_info.c**

4. frees all dynamically allocated memory in a file called **free_dmem.c**

– **You must create a project3.h file that holds the prototypes to these five functions. If you create more than these 5 functions, add those prototypes to this file as well.**

– No variables can be declared outside of a block. That is, no variables can be declared with file scope or as "global".

– You may only use getchar(), printf(), fprintf(), scanf(), fopen() and fclose() for any I/O needs you have while writing this program. Using another other C library I/O program will be a penalty of 50 points.

– You must use pointers within this project such that you have an array of char pointers that contains the addresses of all pizza ingredient strings (the variable that holds the address to this array must be declared as **char ****) and an array for thispizza that contains addresses from the ingredients array (the variable that holds the address to this array must be declared as **char *****).

– Your output must look like the example output below.

– **You cannot use statically declared arrays in this project to store any user data <u>other than the filename in which to store user's information</u>.**

– Your code must work correctly for ANY NUMBER of fresh ingredients (including just 1 and up to the limits of available memory, of course), and these numbers are not known in advance.

– You must use pointers to access user data

– the valgrind program must indicate that your program has no memory leaks

– **You may not access any of the allocated storage space using indexes, as is usually done for a statically declared array, but only by using pointers and pointer arithmetic.**

PROJECT SUBMISSION

Always be sure your linux prompt reflects the correct directory or folder where all of your files to be submitted reside. If you are not in the directory with your files, the following will not work correctly.

You must submit all your project assignments electronically to Carmen in .zip file format. The format of zip command is as follows:

[jones.5684@fl1 project3] $ **zip <zip_filename> <files-to-submit>**

where <zip_filename> is the name of the file you want zip to add all of your files to and <files-to-submit> is a list of the file(s) that make up the project. Remember that you must be at the correct location (the designated directory) for the command to be able to find your files-to-submit.

You must put this command in your Makefile. I highly recommend modifying and using the **verify** command from project1 to ensure that your .zip file has all needful information.

NOTE:

• Your programs MUST be submitted in source code form. Make sure that you zip all the required .c files for the current project (and .h files when necessary), and any other files specified in the assignment description. Do NOT submit the object files (.o) and/or the executable. The grader will not use executables that you submit anyway. She or he will always build/compile your code using your Makefile after inspecting it to insure it compiles using **gcc -std=c99 -pedantic -Wimplicit-function-declaration -Wreturn-type -Wformat** and then test the executable generated by that command.

• It is YOUR responsibility to make sure your code can compile and run on CSE department server **coelinux.cse.ohio-state.edu,** using **gcc -std=c99 -pedantic -Wimplicit-function-declaration -Wreturn-type -Wformat** without generating any errors or warnings or segmentation faults, etc. Any program that generates errors or warnings when compiled or does not run without system errors will receive 0 points. No exceptions!

See below for a sample input to the program. Your program must work for all valid input not just this one. Be creative with your test data!!! ☺

Sample Screen Input/Output for this project for the employee input part: (user provided input is in green.)

```
How many available pizza ingredients do we have today? 10
Enter the 10 ingredients one to a line:
Cheese
Double Cheese
Ham
Pepperoni
Anchovies
Black Olives
Green Olives
Onions
Mushrooms
Green Peppers

Available ingredients today are:
 1. Cheese
 2. Double Cheese
 3. Ham
 4. Pepperoni
 5. Anchovies
 6. Black Olives
 7. Green Olives
 8. Onions
 9. Mushrooms
10. Green Peppers
```

Sample Screen Input/Output for this project for the customer ordering a pizza: (user provided input is in green.)

```
Welcome to our Pizza ordering system!
Today we have the following fresh ingredients available:
 1. Cheese
 2. Double Cheese
 3. Ham
 4. Pepperoni
 5. Anchovies
 6. Black Olives
 7. Green Olives
 8. Onions
 9. Mushrooms
10. Green Peppers

What size pizza would you like to order:
      1)   Small
      2)   Medium
      3)   Large
3
Of our 10 available ingredients, how many do you plan to put on your pizza?  4
Enter the number next to each ingredient you want on your pizza: 2 4 6 9

Here is your receipt:
  Large Pizza                $7.50
 1. Double Cheese            $1.50
 2. Pepperoni                $1.50
 3. Black Olives             $1.50
 4. Mushrooms                $1.50

Total Due:                  $13.50

Do you want to save this receipt? (1=yes, 2=no): 1
What file name do you want to use? mypizza
Today's Receipt has been saved to the file mypizza.
```

Sample File Output for this project (i.e., what is written to disk):

```
     Available Pizza Ingredients today:
      1. Cheese
      2. Double Cheese
      3. Ham
      4. Pepperoni
      5. Anchovies
      6. Black Olives
      7. Green Olives
      8. Onions
      9. Mushrooms
     10. Green Peppers

Here is your receipt:
  Large Pizza                $7.50
 1. Double Cheese            $1.50
 2. Pepperoni                $1.50
 3. Black Olives             $1.50
 4. Mushrooms                $1.50

Total Due:                  $13.50
```