Project 1
100 - 150 lines of code

Sales Analysis

Now that your cash register can calculate change, you need to calculate your total revenue and profits for a given day.

Overview:
Your program will prompt the user for two pieces of information: the input file containing the record of sales for that day and the calculation to perform. You will then open the file and process the data (without storing it in an array) and output the result to the terminal or to a file.

Learning Objectives covered:
1. I can write and compile simple C programs that communicate with the terminal and contain functions, loops and conditionals.
2. I can write a program that adheres to a style guide.
3. I can write high quality documentation.
4. I can write a program that reads from files and writes formatted output to files.

The Input Files:
The sales for each day are stored in a simple text file. The contents of an example file are shown below.

```
1    24    3.50  2.00
2    31    2.50  2.00
3    90    2.50  2.00
4    12    5.75  4.50
```

Each row contains the information for a single item you sell in your store. The columns are as follows:

0: item code
1: Number of items sold
2: Menu price of one item
3: Purchase cost of one item

The calculation:
Your program will prompt the user for the name of the input file and which calculation they want you to perform. The calculations are as follows:
1. Calculate the total revenue for the day. (How much money did customers spend in your shop?)
2. Calculate the total profit for the day. (After subtracting the purchase cost of each item, how much money did you make?)
3. Report a single item. Report on the revenue and profit of a single item

Program structure:
Three code files:

- `analyze_sales.c`
  - `main()`
- `sales_functions.c`
  - Contains at least 3 functions. Examples are shown below. You can use these or create your own. You can decide if the output is printed from the function or from main().
  - `float overall_revenue(FILE *f1):` Calculates the money spent by customers in that day
  - `float overall_profit(FILE *f1):` Calculates profit for that day
  - `void report_one(FILE *f1, int item):` Calculates the revenue and profit for a single item (Ask the user which item in main)
  - Note: I am passing the FILE pointer to the open file to each function, not the file name.
- `sales_functions.h:` Contains the prototypes for the functions in sales_functions.c

Your program will be compiled using the following:

```
gcc analyze_sales.c sales_functions.c -o runSales
```

Your program will be run using the following command:

```
./runSales
```

<u>In main,</u> you should do the following:

Ask the user for the input file and what task to perform using the following format (user input is in bold). Your output must exactly match what is shown below

```
Which program would you like to run:
(1) Calculate overall revenue
(2) Calculate overall profit
(3) Report on a single item
1
Please enter an input file:
Sept_3_2024.txt
```

To read the name, use the following:

```
char name[100];
scanf("%s", name);
```

Check that the option is valid (1, 2 or 3). If not, print the following error message and exit the program:

```
This option is not valid.
```

Otherwise, run the specified task.

If 3 is enter, the following prompts should be used:

```
Which item to analyze?
4
```

The output should have the following format (only one line, based on which function is run. Additional example runs and outputs are found on Canvas.

Option 1:

```
The overall revenue in input_file.txt is $455.50.
```

Option 2:

```
The overall profit in input_file.txt is $344.00.
```

Option 3:

```
Item 4 sold 12 times with a revenue of $69.00 and a profit of $15.00.
```

Restricted material:
You **are allowed** to use arrays in this assignment, although they are not necessary.
You may not use any of the following programming concepts in this assignment:
- structs - global variables - static

Deliverables:

1. Write a document called **README.txt** that summarizes your program, how to run it, and detailing any problems that you had. If you used any outside resources for this assignment, be sure to cite your sources *and* explain in detail how the code works.
2. Your pseudocode, submitted as a .txt file or a picture of hand-written pseudocode.
3. `analyze_sales.c`, `sales_functions.h` and `sales_functions.c`.

Grading:
The full grading rubric is explained in the CS 2303 Project Guidelines document. Five additional rubric items are listed below.

Met	Rubric Item
	Program does not contain restricted items
	Program contains at least 3 functions (plus main)
	Pseudocode contains a section for each function
	Pseudocode outlines the logic of each function
	Function prototypes are in the .h file and it is included correctly in the .c files

The grading breakdown is shown in the following table. To earn a particular grade, you must meet all criteria in the row corresponding to that grade. There are 7 core rubric items and 10 supplemental items (5 in the Rubric document and 5 listed above).

		Rubric Items Met	
	Autograder tests	**Core**	**Supplemental**
Excellent	All Passed	7	9+
Meets Expectations	All Passed	7	8+
Needs Revision	All Passed	N/A	N/A
Not Graded	Failed 1 or More test	N/A	N/A

Notes:

Using your resources:

- If you consult web resources, or other students or staff when developing your program, *you must cite your source*. If you complete the entire program on your own, you should say in your write up file that this is entirely your work.
- If you view an online algorithm, *do not copy it.* Write it out in plain text in your own words, then use those notes to write your own code.