Lab 2

Instructions: Create a basic order processing program which takes a user's order and calculates the tax + total price.

Objectives:

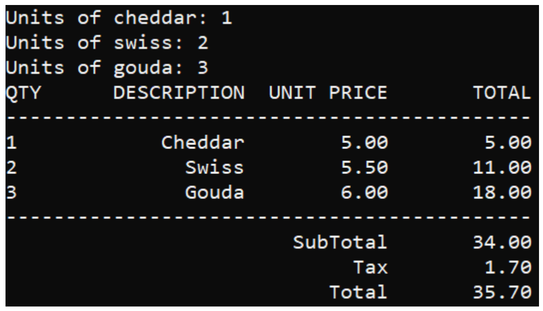
- Further explore input/output with `printf` and `scanf`.
- Use conditionals to determine program behavior based on user inputs.
- Explore file input/output with `fopen`, `fclose`, `fprintf`, and `fscanf`.

Task 1: Cheese orders: A cheese company sells 4 kinds of cheese. The types are as follows:

TYPE	PRICE
Cheddar	$5.00 per unit
Swiss	$5.50 per unit
Gouda	$6.00 per unit

In this program, we will ask the user how many units of each cheese they would like to order. After, we will calculate the total cost, including tax. The order will then be printed to the user as shown in Figure 1.

- Create a .c source file named "`Lab2a.c`".
- In `Lab2a.c`:
  - Use `#define` to define constants for the unit prices of Cheddar, Swiss, and Gouda.
  - Ask the user how many units of Cheddar, Swiss, and Gouda to order.
  - Calculate the total prices for Cheddar, Swiss, and Gouda.
  - Calculate the subtotal by adding the total prices for all three cheeses.
  - Calculate a 5% tax on the subtotal.
  - Calculate the total by adding the tax to the subtotal.
  - Print the order as shown in Figure 1.



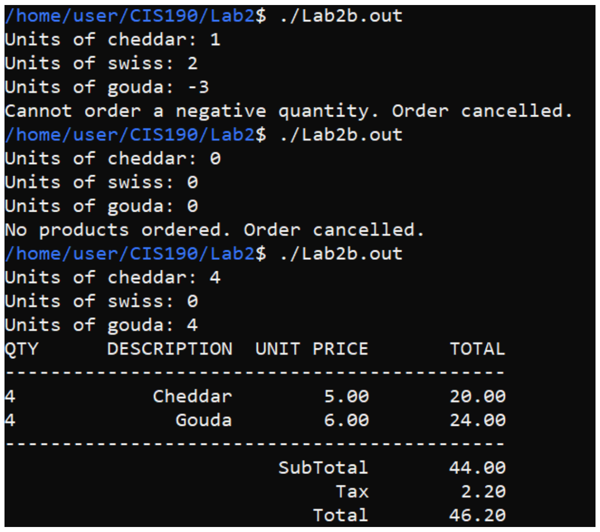
```
Units of cheddar: 1
Units of swiss: 2
Units of gouda: 3
QTY      DESCRIPTION  UNIT PRICE       TOTAL
--------------------------------------------
1            Cheddar        5.00        5.00
2              Swiss        5.50       11.00
3              Gouda        6.00       18.00
--------------------------------------------
                        SubTotal       34.00
                             Tax        1.70
                           Total       35.70
```

Figure 1. Example input/output for `Lab2a.c`.

- Notes on `printf` formatters:
    - "`%d`" inserts an integer; "`%f`" inserts a float; "`%s`" inserts a string.
    - The above "`%`" formatters can be extended to print with padding/justification. Examples:
        - "`%8d`" uses 8 characters to print an integer variable; if the integer is less than 8 digits long, spaces fill the additional spots, and the integer is placed on the *right*.
        - "`%-8d`" uses 8 characters to print an integer variable; if the integer is less than 8 digits long, spaces fill the additional spots, and the integer is placed on the *left*.
        - "`%8.2f`" uses 8 characters to print a float variable up to two decimal places. If the resulting text is less than 8 characters, spaces fill the additional spots, and the float ends up on the right.

Task 2: Extended cheese orders. Starting from `Lab2a.c`, add checks for invalid orders.

- Create a .c source file named "`Lab2b.c`".
- In `Lab2b.c`:
    - Perform all the same steps as in Task 1, with the following additional checks, as shown in Figure 2:
        - If the user orders a negative quantity of any cheese, the order should be cancelled.
        - If the user orders no cheeses, the order should be cancelled.
        - Only cheeses with >0 quantity should appear in the output.

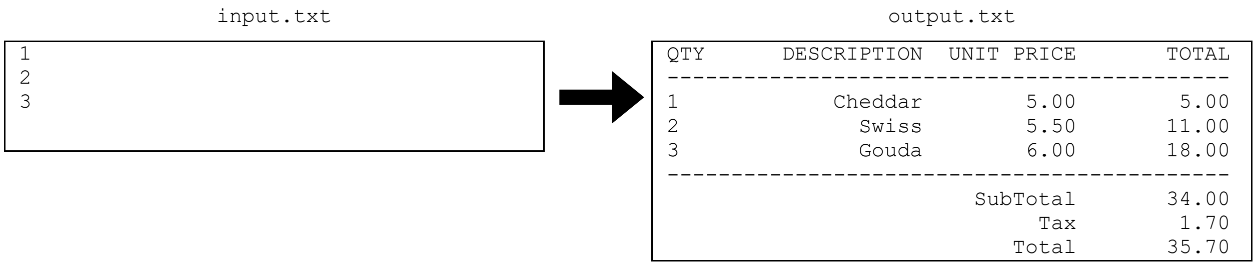


```
/home/user/CIS190/Lab2$ ./Lab2b.out
Units of cheddar: 1
Units of swiss: 2
Units of gouda: -3
Cannot order a negative quantity. Order cancelled.
/home/user/CIS190/Lab2$ ./Lab2b.out
Units of cheddar: 0
Units of swiss: 0
Units of gouda: 0
No products ordered. Order cancelled.
/home/user/CIS190/Lab2$ ./Lab2b.out
Units of cheddar: 4
Units of swiss: 0
Units of gouda: 4
QTY      DESCRIPTION  UNIT PRICE       TOTAL
--------------------------------------------
4            Cheddar        5.00       20.00
4              Gouda        6.00       24.00
--------------------------------------------
                        SubTotal       44.00
                             Tax        2.20
                           Total       46.20
```

Figure 2. Example inputs/outputs for `Lab2b.c`.

Task 3: Cheese orders with file input/output.

- Create a .c source file named "`Lab2c.c`".
- In `Lab2c.c`:
    - Perform all the same steps as in Task 2 with the following modifications:
        - Instead of asking the user for input with `scanf`, obtain inputs from a file called "`input.txt`" using `fopen`, `fscanf`, and `fclose`.
        - Instead of printing the order summary to the terminal with `printf`, print the order to a file called "`output.txt`" using `fopen`, `fprintf`, and `fclose`.
    - Figure 3 shows the `output.txt` that should result from an example `input.txt`:



input.txt

```
1
2
3
```

output.txt

```
QTY      DESCRIPTION  UNIT PRICE        TOTAL
-------------------------------------------
1            Cheddar        5.00         5.00
2              Swiss        5.50        11.00
3              Gouda        6.00        18.00
-------------------------------------------
                        SubTotal        34.00
                             Tax         1.70
                           Total        35.70
```

Figure 3. Example input/output for `Lab2c.c`.

- Hints:
    - Remember to close file pointers with `fclose` before the program closes.
    - On some systems, "`\n`" must be replaced with "`\r\n`" for newlines to properly display in `output.txt`.

Submission details:

- From a terminal in the same directory as Lab1a.c and Lab1b.c, run the following command to produce a compressed archive containing both .c files:

  ```
  $ tar -czvf Lab2_LastName.tar.gz Lab2a.c Lab2b.c Lab2c.c
  ```

  where "LastName" is your last name.

- Submit `Lab2_LastName.tar.gz`.