Understanding the problem:

This program is meant to emulate a pizza ordering system for a restaurant (minus security and convenience obviously). The program should effectively offer two completely different experiences based on whether the user is a customer or employee. The employee should be able to see, sort, and order pizza as well as restaurant info. Employees should be able to edit all aspects including restaurant hours, pizza types and prices. Additionally, the employee should be able to view the order the customers have placed and edit them.

I assume that the files are correctly formatted
I assume the files can be hardcoded in
I assume the user will have some sort of a brain
I assume that program has read write permissions

The program will completely overwrite the files when it ends. When the program starts, it loads all the data from the txt files, it holds all the data until the user decides to quit. When they quit, the program writes everything back to files, overwriting any changes that happened in the in-betweens.

Restaurant Class:

Private structure:

```
private:
   Menu menu;
   employee * employees;
   hours * week;
   string name;
   string phone;
   string address;
   int num_employees;
```

Constructor:

Restaurant()
Constructs restaurant class with defaults values. The pointer will be set to NULL for safety and obviously invalid data will fill the rest of the variables. There will be no constructor that accepts variables and we have a dedicated class function for that.
employees = NULL
week = NULL
name = "NA"
phone = "000-000-0000"
address = "NA"
num_employees = 0

Destructor:
~ Restaurant()
Deletes the dynamically allocated arrays inside the restaurant class.

Delete [] employees
Delete [] week

Accessors:
The following are a bunch of accessors used to print many of the class variables. They are all void, and none accept any input. Since all data
void Restaurant::view_name() print name
void Restaurant::view_address() print address
void Restaurant::view_phone() print phone number
void Restaurant::view_num_employees() print the number of employees
int Restaurant::return_num_employees() return number of employees
void Restaurant::view_employees(){
  for each employee in the array
    print employee info

void search_by_price()
This class should refine the re

Mutators:
The first part of the mutators are very simple, they change a single variable like name or phone.
set_name(string new_name) {name = new_name;}
set_phone(string new_phone) phone = new_phone
set_address(string new_add) address = new_add

Load_data() fetches all the data for the restaurant and employees from files, filling in the classes and structs. It is heavily based on file structure and spacing, using get line and f>> to fetch specifics. It relies on a preprocessor directive for the file name, but will check if the file is valid before opening.
Load_data()
    Create fstream object
    Pass filename and object to verify_file_open which returns an opened object
    Get each line and store in the corresponding variable
    Create array of hours based on number of days open
    Fill hours array
    Close file
    Open with verify employee file, save how many employees there are.
    Create array of employee
    Call get employees

Get_employees(fstream&f)
    Goes the employees file, saving all the info in the array of employees create in load data
Get_employees(fstream&f)
    For each index in the array of employees
        Save id, name, and password

Void place_order()
This function allows the customer to maek

Menu Class:

Private structure:
private:
    int num_pizzas;
    Pizza * pizzas;

Constructor:
Set the pointers to NULL and variables to 0

this-> num_pizzas = 0;
this-> pizzas = NULL;

Accessors:
view_num_pizzas(): {std::cout << num_pizzas; }

view_menu()
Displays the pizzas with their ingredients and pricing.

View_menu()
    For each pizza in pizza array
        Get and print name, price, and ingredients

Void search_menu_by_price()
This function should look for pizzas of a particular price, if a pizza meets the requirements it should be printed out.

Void search_menu_by_price()
    Prompt the user for a price range
    Verify that the input is an integer
    For each pizza in the menu
        Print out pizza if at or below the inputted price

Void search_by_ingredients()
This function when called should allow the user to enter a string for an ingredient and then return any pizzas that contain that ingredient

Void search_by_ingredients()
    for each pizza in the array
        For each ingredient in the pizza
            If ingredient matches input
                Print pizza

Void place_order(Pizza* selection)

After the customer has selected all the pizzas they want, this function should be given the entries and it should store them to file

```
Void place_order(pizza* selection)
        For each pizza in selection
                Place pizza in orders file
```

Mutators:

load_data()

Loads all the data relating to the menu and pizza's. The only file that is accessed is the pizza txt. It will involve opening the file, counting how many pizzas there are, creating an array of that length, then iterating over the file again and storing the various ingredients and name;

```
Load_data()
        Check the file is valid
        If not, re-prompt
        Iterate over the file to count how many pizzas there are
        Create the array of pizza classes
        Iterate over the file for each line
                Store the name, prices, and number of ingredients
                        Call load_ingredients
```

Load_ingredients():

This function is called by the load_data function, its sole purpose is to iterate over the ingredients and store them in an already created array.

```
Load_ingredients():
        Create temporary array of string of length num_ingredients
        For each token in the ingredients list
                Add token to string array
        Pass temp array to fill_ingredients for current pizza
```

Void change_hours()

This function is only available to employees. It prints them the current hours and then offers them the choice to change the hours

```
Void change_hours()
        Prompt user for how many days they would like to change
                For that number of days
                        Ask which day they would like to change and verify
                        Ask new opening hours
                        Ask new closing hours
```

Void add_to_menu()

This function should allow the employee to view and then add a pizza to the list.

```
Void add_to_menu()
```

Prompt user for the name, price, and ingredients for the new pizza
Extend the pizza array by one and add the pizza to the array

void order_from_menu()
This function allows employees to remove an item based on the pizza ID.

Void order_from_menu()
Prompt which pizza the user would like
Check that the ID is an int and below the number of pizzas in the list
Ask the size
Store it in the order structs

Pizza Class:

Private structure:
string name;
int ID;
int small_cost;
int medium_cost;
int large_cost;
int num_ingredients;
string* ingredients;

Constructor:
name = "N/A"
small_cost = -1
medium_cost = -1
large_cost = -1
num_ingredients = -1
ingredients = NULL

Destructor:
Clears the dynamically array
~Pizza()
Delete [] ingredients

Accessors:
Get_name() return name
Get_small_cost() return small_cost
Get_medium_cost() return medium_cost
Get_large_cost() return large cost
Get_num_ingredients() return num_ingredients

Mutators:
Set_name(string new_name) name = new_name
Set_small_cost(int cost) = small_cost = cost
Set_medium_cost(int cost) = medium_cost = cost

Set_large_cost(int cost) = large_cost = cost
Set_num_ingredients(int cost) num_ingredients = num

Create_ingredients_arr(int num_ingredients)
Creates an array of ingredients based on input

```
Create_ingredients_arr(int num_ingredients)
        Ingredients = new string [num_ingredients]
```

Fill_ingredients_arr(int num, string temp[])
Takes a string of ingredients and saves it into the ingredients array of the pizza class

```
Fill_ingredients_arr(int num, string temp[])
        For I less than num
                Ingredients at i = temp at i
```

Non- class functions:

Int get_num_lines(fstream &f)
Takes an open file object and counts how many lines are in the file. This function is useful for finding how many employees are in a file among other things.

```
Int get_num_lines(fstream &f)
        Count =0
        String s
        While not at end of file
                Getline
                Increment count
        Clear buffer
        Go to begging of file
        Return count
```

Void verify_file_open (fstream &f, string filename)
This function takes a closed fstream object and a file name and attempts to open the file. If the file can be opened then the function ends, else it re-prompts for a file until it can be opened.

```
Void verify_file_open(fstream &f, string filename)
        Do
                Open file
                If it is open
                        Break

                Prompt and get new file name

        While(true)
```

void employee_choices()
This function shows the options the employee has. Obviously most of them are not available to the customer.

```
Void employee_choice()
        Print out the options for the user like printing time, add/remove, items
        Take a numbered input
        Verify valid
        Call appropriate function
```

Void get_employees(fstream &f)
This function gets all the employees from a file and places them in an array of employee structs. The filename is a preprocessor directive and the array is already created

```
Void get_employees(fstream &f)
        For I up to number of lines
                f>> id >>first >> last
                save in array
```

Void customer_choice()
This function shows the options available to the customer.

```
Void customer_choice()
        Print out the ability to view menus, search, and order
        Take a numbered input
        Verify valid
        Call appropriate function
```

Bool check_login(int ID, string password)
This function returns a true if login details are correct, else it return false

```
Bool check_login(int ID, string password)
        For each employee in employees array
        If id == id
                If password matches
                        Return true
        Return false
```

Void view_orders()
This function shows the orders that have been placed by the customers

```
Void view_orders()
        For number of orders
                Print order info from struct array
```

Void remove_order()
This function allows employees to remove specific orders

void remove_order()
    ask which order id they would like to remove
    verify id
    if invalid, re-prompt
    remove from structure, fill pointer with NULL

Testing table for choosing user type:

Input	Expected output	Actual output
C	Logged in as customer	
E	Logged in as employee	
e	Logged in as employee	
c	Logged in as customer	
A	Re-prompt	
41	Re-prompt	
Q or q	Exit program	

Testing table for specifying a specific price

Input	Expected output	Actual output
0	Output all pizzas	
-1	Re-prompt	
Ab	Re-prompt	
1293	Output all pizzas below 1293$	
Exit	Re-prompt	

Testing table for specific ingredient

Input	Expected output	Actual output
Bacon	Pizzas with bacon	
123	Pizzas with ingredient 123	
0	Pizzas with ingredient 0	
Av	Pizzas with ingredient Av	
Enter	Continue to wait for input	

Testing table for action selection:

Input	Expected output	Actual output
1 (view menu)	Print menu	
2 (search by cost)	Call search by cost	
Any valid integer between 0 and highest int displayed	Execute that operation	
Basdkbfk	Re-prompt	
Quit	Quit	

Q	Quit	
-123	Re-prompt	
8	Logout	
Help	Re-prompt	

Testing table for changing hours

Input	Expected output	Actual output
M, 1,3	Monday set 1 to 3	
F, 8, 3	Friday set 8 to 3	
Friday, 1, 4	Re-prompt	
1,M, 45	Re-prompt	
Quit	Exit prompt	

Testing table for Adding pizza

Input	Expected output	Actual output
Name, price and ingredient	Accepted and added to array	
Name, price	Accepted, added to array without any ingredient	
Name, ingredients	Re-prompt, require price	
Price, ingredients	Re-prompt, requires name	
1231	Re-prompt, too few arguments	
Duplicate of an existing entry	Ask if they still want to add	
I hat ethis	Re-prompt	

Testing table for removing pizza

Input	Expected output	Actual output
Valid pizza ID	Removed from array	
Invalid ID	Re-prompt	
Pizza name	Re-prompt	