Part 1: Understanding the Problem:
(1 pt) [Rephrase the problem statement here]

To ensure that you understand the problem correctly, answer the following guiding questions:
1. (1 pt) What are the available options if the user logged in as an employee?

2. (1 pt) What are the available options if the user logged in as a customer?

3. (2 pts) In which file(s) is the shop address, phone number, and coffee menu stored? How would your program load/populate the information?

4. (2 pts) What are the required classes for this assignment? List every “has-a” relationship that you’ve observed between the required classes.

5. (1 pt) What is the starting condition of the program? (i.e., what is the starting revenue and number of orders?)

(2 pts) [List assumptions that you are planning to make for this program]
(Hints:
What if the user enters “asdf” when select an option?
What if the user provides a file that does not exist?
What if the user enters a large price smaller than a small price?
What if the user enters a coffee name that already exists in the menu?
Etc.)
Part 2: Program Design:
To help you break the problem down into smaller subtasks, answer the following guiding questions:
1. (1 pt) Of all the available options for both customer and the employee, in what order are you going to implement them and why?

2. (2 pts) The “Big three” (Copy constructor, assignment operator overload, and destructor) is likely needed for a class that contains dynamic memory. In plain English, explain the difference between a copy constructor vs. an assignment operator overload function.

3. (1 pt) Of the required classes, which one(s) would need the Big three implementation and why?

4. (2 pts) Using pseudocode, what is the general algorithm of adding an element into a dynamic array that is already full?

5. (2 pts) Using pseudocode, what is the general algorithm of removing an element from a dynamic array?
In OOP, “Encapsulation” refers to hide the details of implementation of a class from its user. For example, the Coffee class implementation is hidden from the Shop class. If a function needs to access the internal details of a Coffee object, it should be implemented inside the Coffee class.

Consider the following case:
The user wants to print the first coffee object in the menu using a Shop object s.

A common error is to use a chain of accessors/getters, which violates the rule of encapsulation!

Wrong 

```cpp
void Shop::print_first_coffee() {
    cout << “Name: ” << this->m.get_coffee_array()[0].get_name() << endl;
    cout << “Small cost: ” << this->m.get_coffee_arr()[0].get_small();
    ...
}
```

```cpp
Shop s;
s.print_first_coffee();
```

Instead, we should first create a member function in the Coffee class to print a coffee object, such as:

```cpp
void Coffee::print_coffee() {
    cout << “Name: ” << this->name << endl;
    cout << “Small price: ” << this->small_cost << endl;
    ...
}
```

Then, to print the first coffee object in the menu, we can create a function in the Menu class that can directly access the array of Coffee objects, coffee_arr:

```cpp
void Menu::print_a_coffee(int index) {
    // note how print_coffee() is called here
    this->coffee_arr[index].print_coffee();
}
```

Finally, we can call print_a_coffee() from the Shop class:

```cpp
void Shop::print_first_coffee() {
    // Shop class has a Menu object m
    // pass index 0 to print the first coffee object
    this->m.print_a_coffee(0);
}
```

The way of calling it from main() stays the same:

```cpp
Shop s;
s.print_first_coffee();
```
Once understand the rule of encapsulation, use pseudocode to create design of all options. For each option, list what function(s) in which class(es) you are planning to create, and the pseudocode for those functions.

6.1. (1 pt) View shop revenue

6.2. (1 pt) View orders

6.3. (2 pts) View shop details: menu, address, and phone

6.4. (2 pts) Add an item to menu

6.5. (2 pts) Remove an item from menu

6.6. (3 pts) Search by price

6.7. (3 pts) Search by coffee name
6.8. (2 pts) Place an order

7. (1 pt) How would you synchronize the .txt files with your program?

8. (1 pt) After executing once (i.e., complete one option), how would your program display all available options again, until the user choose to quit?

(4 pts) [Use your answer above, create pseudocode (or flowchart) for the remaining functions that you plan to create]

Very importantly, think about how you would connect all pieces together!

*Tip: in order to modularize your program, let each function handle one thing/task. When decompose a program into functions, try listing the verbs/tasks that are performed to solve the problem.*
Part 3: Program Testing
To help you consider the possible test cases, answer the following guiding questions:
1. (1 pt) How would you test if the menu.txt and shop_info.txt are correctly loaded into your program?

2. (1 pt) For your answers in Part 2 Q6, how would you test if the option is working as expected?

3. (1 pt) What inputs need to be error handled?

4. (1 pt) How to ensure that your program does not have any memory leaks? How to locate the memory leaks if there is any?

(6 pts) [Create a testing table that has representative good, bad, and edge cases for each input, and their expected outputs]