# Week 2 Homework: Contact Manager Program

## Task Overview

Create a Python program called contact_manager.py that manages a simple contact list using **dictionaries, functions, and modules**. The program should allow users to add contacts, view all contacts, and calculate the average age of contacts, with functionality split across a main script and a custom module.

## Requirements

1. **Main Script (contact_manager.py):**
    - Use a dictionary to store contacts, where each contact is represented as a nested dictionary with keys: name, age, and phone.
    - Include at least two functions with advanced features (e.g., default arguments, variable-length arguments).
    - Prompt the user to interact with the program (e.g., add a contact, view contacts, or exit).
    - Import and use a custom module (see below).
2. **Custom Module (contact_utils.py):**
    - Create a separate file named contact_utils.py.
    - Define at least one function in this module to process contact data (e.g., calculate the average age of contacts).
    - Import this module into contact_manager.py and use its function.
3. **Specific Tasks:**
    - **Add Contact:** Allow the user to input a name, age, and phone number, then store it in the dictionary.
    - **View Contacts:** Display all contacts in a readable format.
    - **Average Age:** Calculate and display the average age of all contacts using a function from the custom module.
    - **Input Validation:** Ensure age is a positive integer and phone is a string
4. **Bonus (Optional):**
    - Add a feature to search for a contact by name and display their details.

## Submission Guidelines

- Submit two Python files:
    - contact_manager.py
    - contact_utils.py
- Email both files as attachments to the instructor with the subject line: "Python Course - Week 2 Homework - Your Name"
- Due by midnight the day before the Week 3 class

**What You'll Need**

- Python 3.8 or newer.
- Files must be in the same directory to work with the module import.
- Use concepts from Week 2: dictionaries, advanced functions, and modules.

---

## Detailed Instructions

### Step 1: Set Up the Contact Dictionary

- In contact_manager.py, initialize an empty dictionary to store contacts.
- Each contact should be a key-value pair where the key is the contact's name (string) and the value is another dictionary with age (integer) and phone (string).

### Step 2: Create Functions in contact_manager.py

- Write a function to add a contact with at least one advanced feature (e.g., default argument for phone number).
- Write a function to display all contacts, possibly using variable-length arguments to handle flexible output formatting.

### Step 3: Create the Custom Module (contact_utils.py)

- Define a function to calculate the average age of contacts based on the dictionary passed to it.
- Ensure it handles cases where the dictionary is empty (return 0 or a message).

### Step 4: Implement User Interaction

- Use a loop to repeatedly prompt the user for actions (e.g., "Add", "View", "Average", "Exit").
- Validate inputs to ensure the program doesn't crash (e.g., check that age is a number).

### Step 5: Test Your Program

- Add at least 3 contacts, view them, and check the average age.
- Ensure the module function works correctly when imported.

---

## Starter Code

Below is a starter structure to guide students without giving the full solution.

```
# contact_manager.py (rename to week2_LASTNAME_homework.py for submission)
contacts = {}

def add_contact(name, age, phone="Unknown"):
    # Add the contact to the dictionary
    pass

def view_contacts(*args):
    # Display all contacts
    pass

# Import your custom module here
import contact_utils

# Main program loop
while True:
    action = input("Enter action (Add/View/Average/Exit): ").lower()
    if action == "exit":
        break
    # Add your logic here

# contact_utils.py
def calculate_average_age(contact_dict):
    # Calculate and return the average age of contacts
    pass
```

---

## Example Output (For Your Reference)

Here's what a working program might look like when run, to help you evaluate submissions:

```
Enter action (Add/View/Average/Exit): Add
Enter name: Alex
Enter age: 25
Enter phone: 123-456-7890
Contact added!

Enter action (Add/View/Average/Exit): Add
Enter name: Bella
Enter age: 30
Enter phone: 987-654-3210
Contact added!

Enter action (Add/View/Average/Exit): View
Contacts:
- Alex: Age 25, Phone 123-456-7890
- Bella: Age 30, Phone 987-654-3210

Enter action (Add/View/Average/Exit): Average
Average age of contacts: 27.5

Enter action (Add/View/Average/Exit): Exit
Goodbye!
```