 **Task: Contact Book**

* **Description**: Create a program that manages a simple contact book. Users can add a contact (name and phone number), view all contacts, and search for a contact by name.
* **Requirements**:
  + Store contacts in a list of dictionaries (e.g., {"name": "Alice", "phone": "1234567890"}).
  + Write functions for:
    - Adding a contact (prompt for name and phone).
    - Displaying all contacts.
    - Searching for a contact by name (case-insensitive).
  + Validate phone numbers (e.g., must be numeric, 10 digits).
  + Use a while loop for a menu-driven interface (options: add, view, search, exit).
* **Tested Skills**: Lists, dictionaries, functions, user input, string manipulation, loops.

 **Task: Expense Tracker**

* **Description**: Build a program to track daily expenses. Users can add expenses (description and amount) and view the total spent.
* **Requirements**:
  + Store expenses in a list of tuples (e.g., ("Coffee", 5.50)).
  + Write functions for:
    - Adding an expense (prompt for description and amount).
    - Calculating and displaying the total of all expenses.
    - Displaying all expenses.
  + Validate that the amount is a positive number using try-except.
  + Use a menu-driven interface (options: add, view, total, exit).
* **Tested Skills**: Lists, tuples, functions, error handling, user input, basic calculations.

 **Task: File-Based To-Do List**

* **Description**: Write a program that lets users add and view to-do items, saving them to a file so they persist between runs.
* **Requirements**:
  + Store to-do items as a list of strings.
  + Write functions for:
    - Adding a to-do item.
    - Reading and displaying all to-do items from a file (todos.txt).
    - Saving the to-do list to the file after each addition.
  + Handle file errors (e.g., file not found) using try-except.
  + Use a simple command-line interface (options: add, view, exit).
* **Tested Skills**: File I/O, lists, functions, error handling, user input.

 **Task: Simple Calculator Class**

* **Description**: Create a Python class for a basic calculator that performs addition, subtraction, multiplication, and division.
* **Requirements**:
  + Define a Calculator class with methods for each operation.
  + Each method takes two numbers as arguments and returns the result.
  + Include input validation to prevent division by zero using try-except.
  + Write a main program that:
    - Creates a Calculator object.
    - Prompts the user to choose an operation and input two numbers.
    - Displays the result and loops until the user exits.
* **Tested Skills**: Classes, methods, error handling, user input, loops.

 **Task: Unique Word Lister**

* **Description**: Write a program that takes a sentence from the user, splits it into words, and displays only the unique words in alphabetical order.
* **Requirements**:
  + Prompt the user for a sentence.
  + Split the sentence into words, ignoring case and punctuation.
  + Use a set to store unique words.
  + Sort the unique words alphabetically and display them.
  + Write a function to process the sentence and return the unique words.
  + Handle empty input gracefully.