List Organizer – Project #2

Project Overview

In this assignment, you will create a **List Organizer**, a program that helps users manage their lists efficiently. You will implement features to **add, remove, and sort items** while practicing fundamental programming concepts, including **lists, loops, conditionals, functions, and user input handling**. Imagine you are creating a shopping list, packing list, to-do list, etc.

Due Date: *February 21, 2025* (Submit on Google Classroom)

Learning Objectives

By completing this project, you will:
- Use **lists** to store and manipulate data.
- Implement **loops** to maintain program flow.
- Apply **conditionals** to handle user choices and errors.
- Handle **user input** effectively to prevent errors.
- Structure code into **functions** for better organization.

Project Requirements

1. Create a List

- Use a **list** to store items.
- Allow users to **add and remove** items dynamically.

2. Build a Main Menu

- Implement a **looping menu** that continuously displays options until the user chooses to exit.
- Include the following options:
    1. **Add Items**
    2. **Remove an Item**
    3. **View List**
    4. **Sort List Alphabetically**
    5. **Exit the Program**

3. Implement Adding Items

- Allow users to input **multiple items** before returning to the main menu.

- Ensure **consistent formatting** (e.g., capitalize items).
- Prevent users from adding blank entries.

4. Implement Item Removal

- Display the **current list** with numbered options.
- Allow users to remove an item by entering its **position number**.
- Handle **invalid inputs** (e.g., non-numeric entries or numbers out of range).

5. Implement Sorting

- Provide an option to **sort the list alphabetically**.
- Notify users that the list has been sorted.

6. Handle Exiting the Program

- Ensure the program exits cleanly when the user selects **Exit**.
- Print a **farewell message** before terminating.

7. Error Handling and Validation

- Prevent **crashes** by handling unexpected inputs (e.g., letters when a number is expected).
- Ensure the program works even if the list is **empty** (e.g., prevent removal when no items exist).

Bonus Challenges *(Optional)*

- Add an option to **edit an existing item** instead of removing and re-adding it.
- Implement **search functionality** to check if an item is already in the list.
- Save the list to a **text file** so it persists after the program closes.
