README – Pokémon Card Collection Tracker

1. Overview

The Pokémon Card Collection Tracker is a C++ console program that helps users manage a small personal inventory of Pokémon trading cards.

The program allows users to:

- **Add new Pokémon cards**
- **Remove cards**
- **Search cards by name**
- **Sort the collection by card value (using Bubble Sort)**
  Display all stored cards

This project demonstrates structured data management, searching, sorting, and user-driven menu operations.

---

2. Design Decisions

Structured Data (Structs)

I used a struct named PokemonCard to group:

- **card name**
- **Type of pokemon**
- **Cost of pokemon**
- **The hp of pokemon**

Using a struct makes the data organized and easier to pass to functions.

Programming Constructs Used

The project uses:

- **arrays**

- **loops**
- **conditionals**
- **functions**
- **structs**
- **input validation**

These are required components of the assignment and help keep the code modular and readable.

Searching Method

- **Linear Search**
  I used linear search because:
    - **the list is small**
    - **it is simple to implement**
    - **sorting is not required before searching**
      It scans each element until the card is found.

Sorting Method

- **Bubble Sort**
- **bubble sort is easy to implement**
- **the dataset is small, so performance is not an issue**
  The sort organizes the cards by value from lowest to highest.

Alternative Approaches Considered

I considered:

- **selection sort**
- **binary search**
- **file storage**

I did not use them because:

- **bubble sort was sufficient**
- **I didn't need to use any file when the user can implement the cards themselves**

---

3. Testing Summary

Test Case ID	Description	Input	Expected Output	Actual Output	Pass/Fail
TC-01	Add a new Pokémon card	Name: *Pikachu*Type: *Electric*Cost: *5.50*HP: *60*	"Card Successfully Added!"	"Card Successfully Added!"	P
TC-02	Add another card	Name: *Charizard*Type: *Fire*Cost: *12.00*HP: *150*	"Card Successfully Added!"	"Card Successfully Added!"	P
TC-03	Display all cards	Option 2	Shows full list: 1. Pikachu …2. Charizard …	List printed correctly	P
TC-04	Search for existing card	Search: *Pikachu*	Shows Pikachu card info	Shows Pikachu card info	P
TC-05	Search for non-existent card	Search: *Mewtwo*	"Card not found."	"Card not found."	P
TC-06	Remove existing card	Remove: *Pikachu*	"Card removed."	"Card removed."	P

TC-07	**Remove non-existent card**	**Remove: *Bulbasaur***	**“Card not found.”**	**“Card not found.”**	**P**
TC-08	**Sort cards by value using bubble sort**	**Before sort:Charizard: $12.00Squirtle: $4.00Onix: $7.00**	**Sorted:Squirtle ($4.00)Onix ($7.00)Charizard ($12.00)**	**Sorted correctly**	**P**
TC-09	**Invalid input for cost**	**Enter Cost: *ABC***	**Should reject input and not crash**	**Program stops reading + loops after invalid input1**	**P**
TC-10	**Edge Case: empty string search**	**Search: ""**	**“Card not found.”**	**“Card not found.”**	**P**

Testing Methods Used

- **Manual testing for each menu option**
- **Edge-case testing (empty list, full list, incorrect inputs)**
- **Repeated add/remove cycles to check stability**

Example Invalid Inputs

- **Entering letters for value → program re-prompts**
- **Removing card when list is empty → prints error455**
- **Adding card when list is full → prints warning**

---

4. Technical Walkthrough

Main Functionalities

- **addCard() – adds a new Pokémon card to the array**
  removeCard() – removes card by searching its name
- **displayCards() – prints all stored cards**
- **linearSearch() – finds card index by name**
  bubbleSort() – sorts by value (ascending)
- **displayMenu() – shows user options**

Program Flow

1. **Program displays menu**
2. **User selects an action**
3. **Program executes function**
4. **Menu repeats until user quits**

Video Demonstration

https://youtu.be/ouZq2vbKnck

---

5. Challenges and Lessons Learned

Challenges

- **Designing the menu loop and preventing input crashes**
- **Deciding between sorting algorithms**
- **Managing array size limits**
- **Removing elements by shifting the array**

Lessons Learned

- **Structs make data cleaner and easier to manage**
- **Sorting and searching algorithms are useful in real programs**
- **Planning the program flow saves time in debugging**

---

6. Future Improvements

If I had more time, I would add:

- **File storage to save data between runs**
- **Binary search for faster lookups**
- **Ability to edit card details**