CSF101- Programming Methodology

Assignment 2 (15 Marks)

Instruction:

This assignment has two parts. You are required to submit your work in a GitHub repository with the following naming structure (folder name): StudentName_studentNumber_A2

Eg: DarshanSubedi_02190108_A2

Additionally, you are required to submit your work in a GitHub repository with the following naming structure (folder name): StudentName_studentNumber_A2

Eg: DarshanSubedi_02190108_A2

Your Python file submissions must be inside the above-mentioned repository. For each part, you must have **two separate Python files**, their names must be as follows:

StudentName_studentNumber_A2_PA.py (for Part A)
StudentName_studentNumber_A2_PB.py (for Part B)

Eg:
DarshanSubedi_02190108_A2_PA.py
DarshanSubedi_02190108_A2_PB.py

Please also submit a report in .doc format for assignment grading on VLE with your complete python code and include screenshots of the outputs for each part in the assignment.

Your report assignment will be subject to Turnitin plagiarism checker, and your GitHub code submissions will be checked for functionality.

Part A: *More Games (5 Marks)*

Create a command-line program in a single Python file that implements 5 mathematical and string-processing functions. The program should:

Program Flow

1. Display a menu showing the available functions (1-5)
2. Prompt the user to select a function by entering a number from 1-5
3. Based on the selection, prompt for and collect the required input
4. Execute the selected function and display the result
5. Allow the user to continue with another calculation or exit

Required Functions
1. Guess number game
2. Rock paper scissors game
3. Trivia pursuit quiz game with different categories and multiple choice.
4. Pokemon Card Binder Manager
5. Overall scoring system

1. Guess number game
Write a guess number game for a range of numbers.
- Input: User guesses
- Output: User guesses validated to be correct or not
- Score: Number of valid numbers - number of guesses with minimum score of 0

2. Rock paper scissors game
Write a text-based rock paper scissors game with a randomized computer opponent.
- Input: User guesses
- Output: Information on user round win or loss
- Score: Number of wins versus computer

3. Trivia Pursuit Quiz Game
Write a trivia pursuit game with questions grouped in different categories and multiple choice selection for answers
- Input: User multiple choice guesses
- Output: User guesses validated to be correct or not
- Score: Number of correct guesses

4. Pokemon Card Binder Manager
Call the Pokemon Python Card Game is described in Part B

Technical Requirements
1. All input validation must be implemented
2. Invalid inputs should be handled gracefully with appropriate error messages
3. Each function should be properly documented with docstrings
4. The program should continue running until the user chooses to exit

Sample Program Output

```
Select a function (0-5):
1. Guess Number game
2. Rock paper scissors game
3. Trivia Pursuit Game
4. Pokemon Card Binder Manager
5. Check Current Overall score
0. Exit program

Enter your choice: 4
Welcome to Pokemon Card Binder Manager!

Main Menu:
```

```
1. Add Pokemon card
2. Reset binder
3. View current placements
4. Exit
```

Part B: *Pokemon Card Binder Manager (10 Marks)*

Background

A Pokemon card collector known as Deep Pocket Monster (DPM) needs a program to organize his complete Pokemon card collection. The collection needs to be organized based on Pokedex numbers in a binder where:
- Each binder page holds 64 cards (8x8 grid)
- Cards must be placed sequentially based on Pokedex numbers
- No duplicate cards are allowed
- Card placements must persist within the current program session
- Scoring would be the number of Pokemon card in the binder upon exit of the program

Program Requirements

Input
- Select Mode from the Main Menu
    - For Mode 1:
        - Pokedex number (integer between 1 and the maximum Pokedex number which is 1025)
    - For Mode 2:
        - "CONFIRM" if the user is looking to reset the binder and "EXIT" if the user is looking to exit the main menu
    - For Mode 3:
        - No input is required
    - For Mode 4:
        - No input is required

Output
- For Mode 1:
    - Page number
    - Position: position of the Pokemon card on the binder
    - Status: if the Pokemon card is newly added to the binder or if it already exists
- For Mode 2:
    - A warning message for reset and a prompt to type either CONFIRM or EXIT
- For Mode 3:
    - Current cards in the binder
    - Total cards in the binder
    - Percentage completion of the binder
- For Mode 4:
    - End session message

Technical Requirements

1. In-Memory Storage
   - Maintain card placements in an appropriate data structure during program runtime
   - The program should track: The Pokedex number, page number, and coordinates
   - Data persists only within the current session
   - However, in its default state, the program resumes from the previous session unless the user decides to reset the session

2. Card Placement Logic
   - Cards must be placed sequentially (lower Pokedex numbers before higher ones)
   - The first page starts with Pokedex #1 at position (1,1)
   - Each page fills left-to-right, top-to-bottom
   - The new page starts when the current page reaches 64 cards

3. Validation
   - Check for invalid Pokedex numbers
   - Prevent duplicate entries

4. Collection Completion
   - Track the total number of unique Pokemon added
   - Display "You have caught them all!!" when collection is complete

5. Session Management
   - Include a reset option in the main menu
   - Reset should:
      - Clear all data from the in-memory storage
      - Requires explicit confirmation to prevent accidental resets
      - Provides clear feedback after reset

Sample Program Flow

```
Welcome to Pokemon Card Binder Manager!

Main Menu:
1. Add Pokemon card
2. Reset binder
3. View current placements
4. Exit

Select option: 1
Enter Pokedex number: 25

Output:
Page: 1
Position: Row 4, Column 1
Status: Added Pokedex #25 to binder
```

```
Select option: 1
Enter Pokedex number: 151

Output:
Page: 3
Position: Row 3, Column 7
Status: Added Pokedex #151 to binder

Select option: 3
Current Binder Contents:
------------------------
Pokedex #25:
  Page: 1
  Position: Row 4, Column 1

Pokedex #151:
  Page: 3
  Position: Row 3, Column 7
------------------------
Total cards in binder: 2
% completion: 0.2%

Select option: 2
WARNING: This will delete ALL Pokemon cards from the binder.
This action cannot be undone.
Type 'CONFIRM' to reset or 'EXIT' to return to the Main Menu: CONFIRM
The binder reset was successful! All cards have been removed.

Select option: 3
Current Binder Contents:
------------------------
The binder is empty.

Total cards in binder: 0
% completion: 0%

Select option: 1
Enter Pokedex number: 25
Output:
Page: 1
Position: Row 1, Column 1
Status: Added Pokedex #25 to binder

Select option: 4
Thank you for using Pokemon Card Binder Manager!
```

Implementation Guidelines

1. Use appropriate data structures for efficient lookup and storage

2. Include error handling for all possible edge cases
3. Add comments explaining the logic and algorithms used

Technical Details to Include in Comments
1. Explain the algorithm for calculating page numbers
2. Document the coordinate system implementation
3. Describe the chosen in-memory data structure and its operations
4. Explain any additional data structures chosen and why
5. Document the reset implementation
6. Explain the data deletion process