Assignment 1: Advanced Palindrome Check Server-Client Application (10%)

Start Date: 29th January 2025
Submission Deadline: 14th February 2025 (11.59 PM)
Submission Medium: GradeScope

Background

Mochi the panda 🐼 has a special talent for recognizing patterns, especially when it comes to palindromes! In this assignment, you'll step into Mochi's world and help build a server-client application using socket programming to detect palindromes. Not only will you assist Mochi in spotting simple palindromes that read the same forward and backward, but you'll also tackle exciting challenges, like determining if a string can be rearranged to form a palindrome. By joining Mochi on this adventure, you'll sharpen your skills in algorithm design, concurrent programming, and system architecture.

Objective

The objective is to create a server-client application utilizing socket programming in Python. The application should offer the following functionalities:
1. **Simple Palindrome Check:** The server should check if the input string is a palindrome, ignoring case, spaces, punctuation, and special characters.
2. **Complex Palindrome Check:** The server should determine if the string can be rearranged to form a palindrome. Additionally, compute the complexity score based on the minimum number of character swaps needed to convert the string into a palindrome.
3. **Concurrency:** Handle multiple clients simultaneously using multithreading or asynchronous programming.

Assignment Details

Core Functionalities

1. Simple Palindrome Check:
- The server must determine if a given string is a palindrome.
- The check must be case-insensitive and ignore spaces, punctuation, and special char
    - Example:
      Input: "A man, a plan, a canal, Panama"
      Process: Ignore spaces, punctuation, and case sensitivity. The string should be treated as "amanaplanacanalpanama".
      Output: True (The string is a palindrome) acters.

2. Complex Palindrome Check:
- The server should check if the string can be rearranged to form a palindrome.
- A complexity score should be calculated, indicating the minimum number of swaps required to convert the string into a palindrome.

- Example without Complexity Score:
  Input: "civic"

Process: Check if the string can be rearranged into a palindrome.
The string "civic" already forms a palindrome without rearrangement.
Output: True (The string is already a palindrome)

- Example with Complexity Score:
  Input: "ivicc"
  Process: Rearrange the string into a palindrome by swapping characters.
  2 swaps ("ivicc" -> "civic") is required.
  Output:
  Can form a palindrome: True
  Complexity score: 2 (number of swaps)

Technical Requirements

Server Requirements

1. Socket Programming:
   - The server must use TCP sockets.
   - Handle multiple clients using multithreading or asynchronous programming.
2. Logging:
   - Log all client requests, including:
     - Client IP address.
     - Requested check type (simple/complex).
     - Input string.
     - Result (true/false) and complexity score (if applicable).
3. Services:
   - Provide two services:
     - Simple Palindrome Check
     - Complex Palindrome Check

Client Requirements

1. Socket Communication:
   - The client should connect to the server using TCP sockets.
2. Menu:
   - The client should offer a menu to select between simple or complex palindrome checks.
3. Input to Server:
   - The client should send the user-provided input to the server for validation.
4. Timeout Handling:
   - Implement a timeout mechanism where if the server does not respond within 5 seconds, the client should display a timeout error and retry the connection up to 3 times before exiting.

Input Constraints

- The server should handle strings containing mixed case, spaces, special characters, and numbers.
- Non-alphanumeric characters should be ignored in all checks.

Code and Documentation

1. Code Organization:
   - Organize your code into modular, reusable functions.
   - Provide comments explaining the key parts of your code.
2. README File:
   - Include a README file with:
     - Instructions for compiling and running the server and client.
     - Example inputs and outputs.
     - Assumptions and limitations.

Additional Features (Twists)

1. Timeout and Retry Mechanism (Client):
   - The client must handle server timeouts gracefully and retry up to 3 times.
2. Logging (Server):
   - Maintain a log file that tracks all requests in a structured format.

Extra Credit (Optional, 2 Points)

1. Simple Task for Extra Credit:
   - Implement one of the following features for 2 bonus points (these points are excluded from the total grade of 30 marks):
     - Basic Encryption: Implement basic encryption for data transmitted between the server and client (e.g., using a simple cipher).
     - Graphical User Interface (GUI) for Client: Develop a simple GUI using libraries like tkinter or PyQt to enhance user interaction.

Evaluation Criteria (Grading Rubric)

The assignment will be graded out of 30 points, with an additional 2 bonus points available for extra credit.

1. Core Functionality (15 points):
- Simple Palindrome Check + Demo: 6 points
- Complex Palindrome Check + Demo: 7 points
- Concurrency (handling multiple clients): 2 points

2. Additional Features (8 points):
- Timeout Mechanism: 3 points
- Logging: 2 points
- Input Constraints: 3 points

3. Code Quality and Documentation (7 points):
- Code Modularity: 2 points
- Code Comments and Readability: 2 points
- Documentation: 3 points

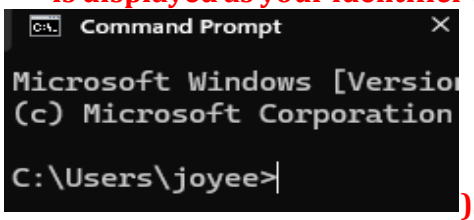
4. Extra Credit (Optional, 2 points):
- Simple Encryption **or** GUI Client: 2 points (excluded from the total of 30 points).

Submission Guidelines

1. Code Submission: Submit the complete source code for the server and client with appropriate comments in Gradescope.
2. Demo Submission: Submit a screen recording in GradeScope.
3. Documentation: Submit a README file in GradeScope explaining how to compile and run the program, example inputs and outputs, and assumptions/design decisions.
4. Server Log File: Submit a sample log file generated during the program's execution in GradeScope.
5. Extra Credit (if implemented): Provide details about encryption or GUI design in the README file.

Deliverables

1. Fully functional server and client applications (.py).
2. A sample log file generated by the server.
3. A short report (README) detailing your implementation, challenges faced, and any extra features added.
4. A video demo of the advanced palindrome checker in action in GradeScope **(Your code should be executed in the command prompt, ensuring that your computer name is displayed as your identifier to verify that the work is genuinely yours:**



).