2xc3 Final Project, C programming- Password Resilience Analyzer

In the realm of cybersecurity, the strength of passwords is crucial in defending sensitive information from unauthorized incursions and potential security breaches. The **Password Resilience Analyzer** has emerged as an essential tool in this arena, allowing both individuals and organizations to assess the resilience of their passwords against the onslaught of brute-force and sophisticated attacks. By mimicking possible attack strategies on a selected password, the Resilience analyzer offers a deeper understanding of the potential vulnerabilities inherent in a password, prompting users to develop stronger, more secure combinations. As cyber threats continue to advance in complexity, ensuring that passwords are fortified against these challenges is of utmost importance, and tools like the **Password Resilience Analyzer** are indispensable allies in this dynamic digital theater.

Objective: Your task is to design and develop a software program using the **C language**, aptly named the "**Password Resilience Analyzer**". This program will be tasked with assessing the strength of a password, bearing in mind a set of predetermined rules and guidelines.

Background: In today's digitally-driven age, a strong password is more than just a gatekeeper to personal accounts—it's a fortress that guards sensitive data and ensures digital security. But, what really defines the strength of a password? It's not merely about length or complexity; it's about understanding a mix of characteristics that withstand various hacking techniques. Factors like **length**, **use of diverse character sets**, **avoidance of predictable patterns**, and the **non-reliance** on **easily guessable information all come into play**. With cyber threats becoming increasingly sophisticated, the importance of robust passwords cannot be overstated. Through the **Password Resilience Analyzer**, you'll delve deep into the anatomy of passwords, examining specific traits and characteristics to determine their robustness. By the end of this project, you'll not only master the art of creating strong passwords but also be equipped with the skills to help others understand and appreciate the nuances of password security.

Deliverable features & requirements of the project (24 Marks):

1. Main Menu (3 Marks)

Description: Implement a user-friendly main menu interface to navigate through the program.

- Display the main interface: 1 Mark
- Handle user input for option selections correctly: 1 Mark
- Provide accurate responses to each selection: 1 Mark

Options to provide:

- ***Test a new password*****:** Allows the user to enter a password for strength analysis.
- ***View strength of the last tested password***: Shows the strength rating (Weak, Moderate, Strong) of the previously tested password.
- ***Exit***: Close the program.

2. **Password Input** (4 Marks)**:**

Prompt the user to input a password and store it for analysis.

- Prompt the user to enter a password for testing. 1 Mark
- Use an array (essentially a string in C) to store the entered password. 1 Mark

- If the entered password is less than 8 characters, display an error message and ask the user to re-enter a longer password. 2 Marks

3. **Strength Tests** (7 Marks):

A password's strength is determined based on multiple criteria. For each criterion met, the password gains a strength point.

Description: Evaluate the password based on various criteria.

- Check for the presence of lowercase letters: 1.5 Mark
- Check for the presence of uppercase letters: 1.5 Mark
- Check for numerical digits: 1 Mark
- Check for special characters such as **!@#$%^&*()**: 1.5 Mark
- Award Mark for lengthy passwords (12+ characters): 1.5 Mark

4. **Functions to Implement (**5 Marks**):**

Each criterion above should have a dedicated function to check its condition.

You'll need to create the following functions:

- Implement **hasLowercase()**: Does the password contain lowercase letters? 1 Mark
- Implement **hasUppercase()**: Does the password have uppercase letters? 1 Mark
- Implement **hasDigit()**: Does the password contain numbers? 1 Mark
- Implement **hasSpecialChar()**: Are there special characters in the password? 1 Mark
- Implement **evaluateStrength()**: Based on the previous checks, what's the total strength score of the password? 1 Mark

5. **Password Strength Score (2 Marks):**

- The total strength of the password is computed as a score, ranging from **0 (weakest) to 5 (strongest)**. 1 Mark
- As each strength criterion is met, a point is added to the score. For instance, if a password has a digit and a lowercase letter but nothing else, it gets 2 points. 1 Mark

6. **Output (3 Mark):**

- Based on the computed score, the password is categorized into:
  - 0-2 points: "Weak"
  - 3-4 points: "Moderate"
  - 5 points: "Strong"
- Display the strength category to the user after analysis.

Tips:

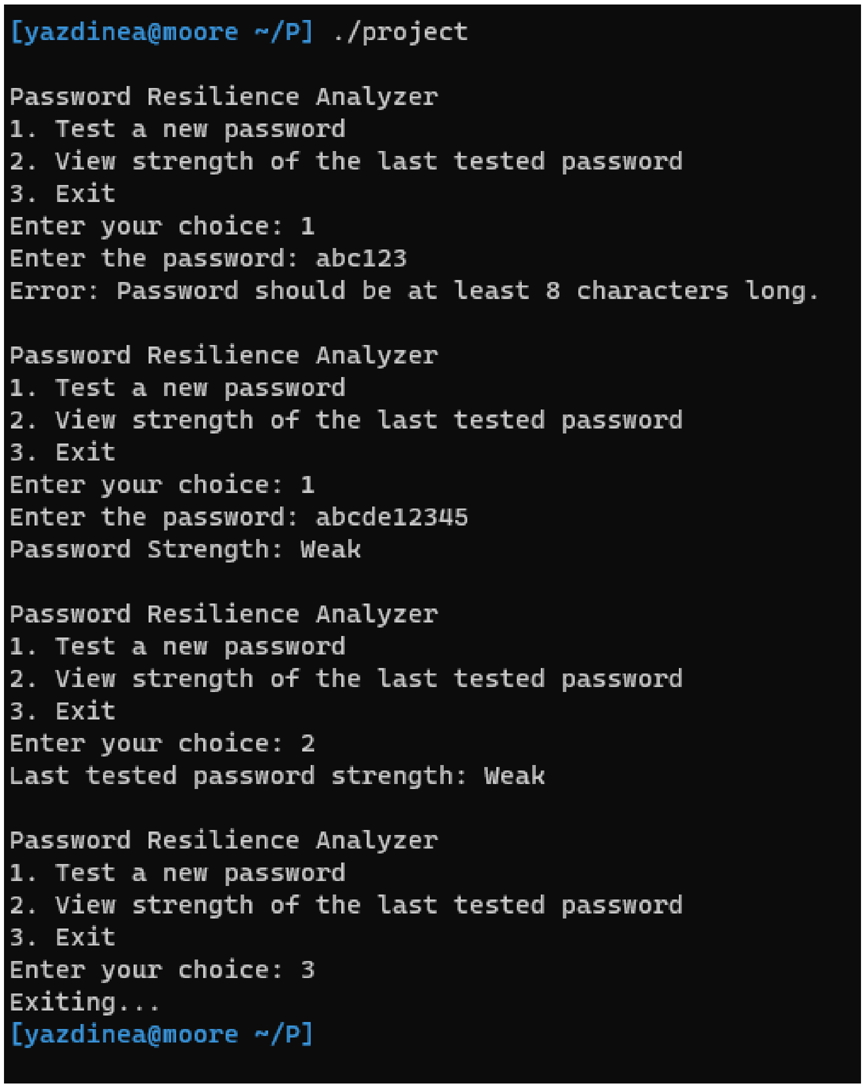
Here are some hints:

1. **Setting up the Structure**:
    - Begin by understanding the flow of the program. Sketching a flowchart or pseudo-code can help in this process.
2. **Main Menu**:
    - Use a loop (like a **while** loop) to continuously display the main menu until the user chooses to exit.
    - Utilize a switch-case or if-else statements to handle the user's choice.
3. **Password Input**:
    - Remember to check the length of the password immediately after the user inputs it. If it's less than the required length, prompt the user to enter it again.
    - You can use the **strlen** function to get the length of a string.
4. **Strength Tests**:
    - For checking the presence of lowercase, uppercase, digits, and special characters, consider iterating through each character of the password string.
    - For checking repeated sequences, think about how you might look for repeated patterns in a string. (Hint: Nested loops might be helpful here.)
    - For analyzing against dictionary words, you might consider a predefined list of words and then see if any of those words exist in the password. (This will be a basic check; in real-world scenarios, this would be far more extensive.)
5. **Functions**:
    - Each function should be singular in its responsibility. For instance, **hasLowercase** should only check for lowercase characters and return a result based on its findings.
    - Remember to pass the password string as an argument to each of these functions.
6. **Password Strength Score**:
    - Initialize the score as 0. For every criterion met, increment the score.
    - It might be helpful to use the results (return values) from the functions to determine the total score.
7. **Output**:
    - Based on the score, display the appropriate strength of the password. You can use if-else statements for this.
8. **General Coding Tips**:
    - Start simple. First, get the basic structure working and then expand upon it.

- Test often. After implementing each function or feature, test it to ensure it works as expected.
- Comment your code. This will not only help you understand your thought process later but will also be beneficial for anyone reading your code.

Test Case 1 Checking:

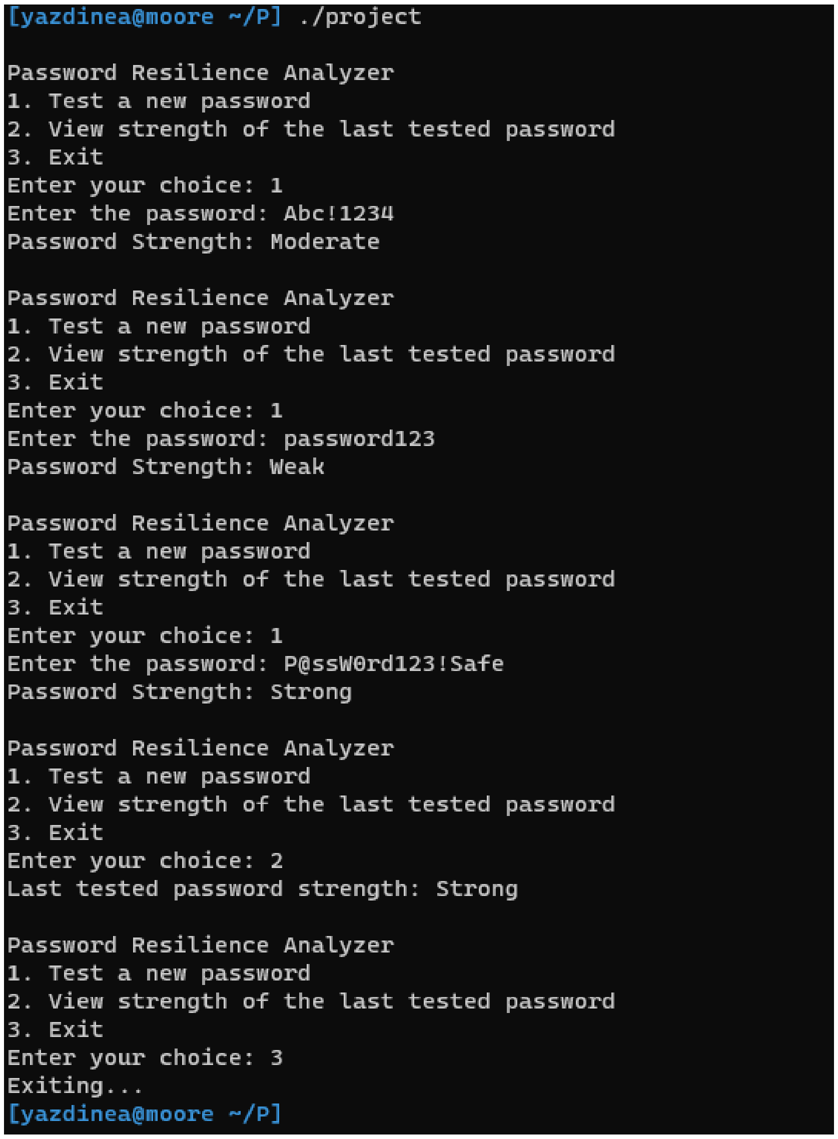
- **Main Menu**
- **Short pass**
- **Weak pass**



```
[yazdinea@moore ~/P] ./project

Password Resilience Analyzer
1. Test a new password
2. View strength of the last tested password
3. Exit
Enter your choice: 1
Enter the password: abc123
Error: Password should be at least 8 characters long.

Password Resilience Analyzer
1. Test a new password
2. View strength of the last tested password
3. Exit
Enter your choice: 1
Enter the password: abcde12345
Password Strength: Weak

Password Resilience Analyzer
1. Test a new password
2. View strength of the last tested password
3. Exit
Enter your choice: 2
Last tested password strength: Weak

Password Resilience Analyzer
1. Test a new password
2. View strength of the last tested password
3. Exit
Enter your choice: 3
Exiting...
[yazdinea@moore ~/P]
```

Test Case 2 checking:

- **Moderate pass**
- **Strong pass**
- **Check the strength of the last tested password**
- **Exit**



```
[yazdinea@moore ~/P] ./project

Password Resilience Analyzer
1. Test a new password
2. View strength of the last tested password
3. Exit
Enter your choice: 1
Enter the password: Abc!1234
Password Strength: Moderate

Password Resilience Analyzer
1. Test a new password
2. View strength of the last tested password
3. Exit
Enter your choice: 1
Enter the password: password123
Password Strength: Weak

Password Resilience Analyzer
1. Test a new password
2. View strength of the last tested password
3. Exit
Enter your choice: 1
Enter the password: P@ssW0rd123!Safe
Password Strength: Strong

Password Resilience Analyzer
1. Test a new password
2. View strength of the last tested password
3. Exit
Enter your choice: 2
Last tested password strength: Strong

Password Resilience Analyzer
1. Test a new password
2. View strength of the last tested password
3. Exit
Enter your choice: 3
Exiting...
[yazdinea@moore ~/P]
```

Evaluation Criteria:

- Code Organization: How modular is your code? Is each function performing a single responsibility?
- Accuracy: Does your program correctly identify password strengths based on the criteria?
- User Experience: Is it easy for the user to understand and navigate through the program?

Submission:

For submission, please provide a C file named in the format "FirstnameLastname.c" (for example, "AAAABBBBB.c"). Additionally, include a PDF file that details the testing of your project's functionality and demonstrates that it meets the required criteria, as shown in the sample runs.

The deadline for the final C project submission is November 28 at 11 P.M. If you wish to submit early, you can do so during your minor lab sessions and receive your grade immediately. For students enrolled in L01, you have the option to submit your project during Makeup Session A on November 27 and promptly receive your grade from the TA. Similarly, students in L02 and L03 can submit their projects on November 28 during their respective registered lab sessions and receive their grades shortly after.

If you deliver the project in person to a TA in the lab session, you must submit it in AVENE as well!