PASSWORD STRENGTH CHECKER

- ABDUL HANAN CH
- ZOHA ASAD
- FIZZAH AMIR
- NAIMAL MUNIR
- EMAN FATIMA

INTRODUCTION

- LENGTH

- CHARACTER VARIETY

- ENTROPY

- COMMON PATTERNS

MATHEMATICAL FOUNDATION OF PASSWORD STRENGTH CHECKER

- Combinatorics

  N = c^L

- Entropy

  H = log (c) base 2

    - Number theory
    - Pattern matching dictionary lookup

PASSWORD STRENGTH CHECKER AND LOGICAL PROOF

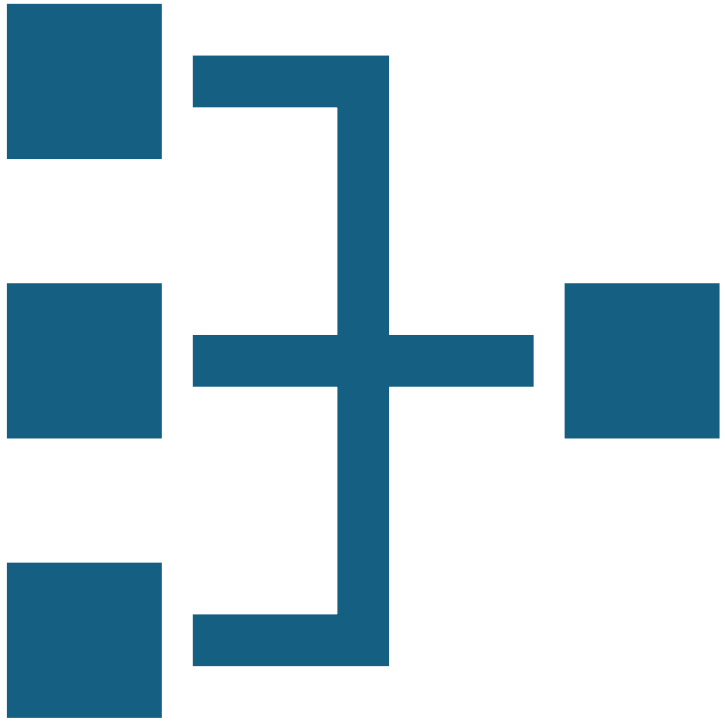
- A strong password must satisfy all conditions, meaning it must include at least one character from each set. This is represented using logical conjunction (∧, "AND").

- A weak password fails to meet one or more of these conditions, represented using logical disjunction (∨, "OR") or negation (¬, "NOT").

PASSWORD STRENGTH CHECKER AND SET THEORY

In set theory, we represent password characters as sets:

- A: Uppercase letters
- B: Lowercase letters
- C: Digits
- D: Special characters



Set Operations:

- Union ∪: Combines characters from either set. E.g., A∪B (uppercase or lowercase letters).
- Intersection (A∩C)=∅ (no overlap between uppercase letters and digits).

- Lattice theory and password strength checker:

  A lattice organizes password criteria into subsets based on the number of conditions met. This enables a structured analysis of password strength.

- Universal Set U: Includes all allowed characters:Uppercase (A), Lowercase (B), Digits (C), Special Characters (D)
- Intersection for Strong Passwords:
- P(strong)=$A \cap B \cap C \cap D$
- p(weak)=U-(A∩b∩c∩d)

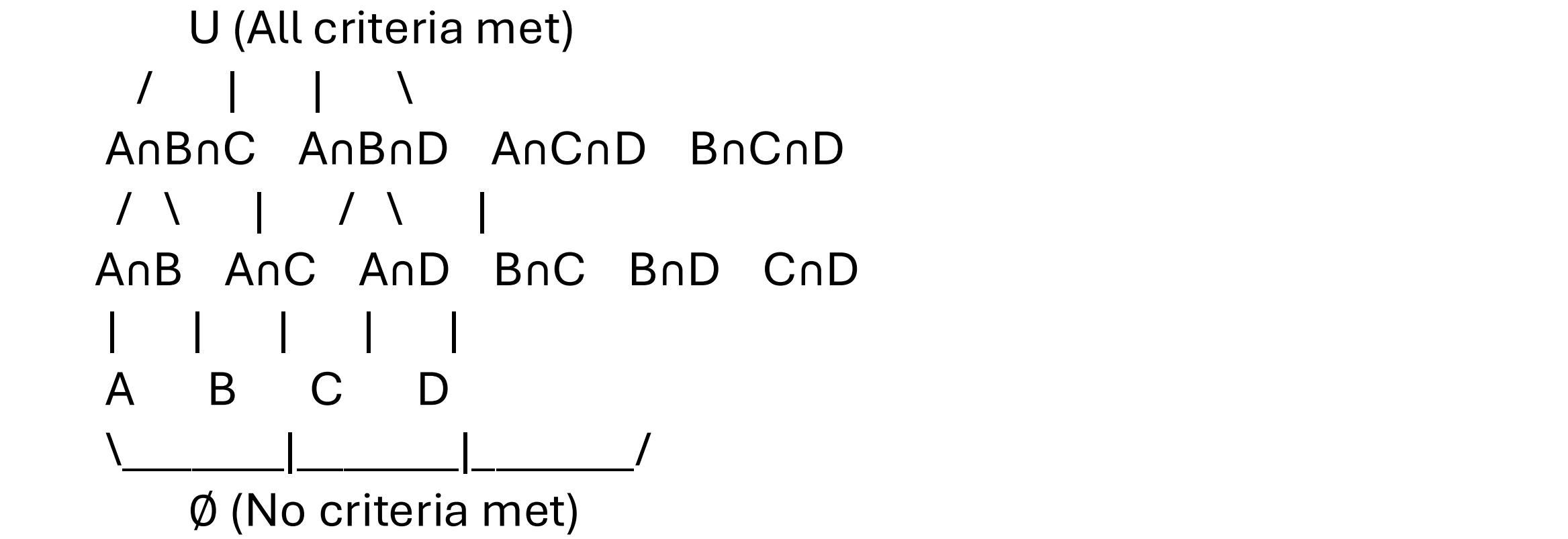
Lattice Visualization:

Top Node(u):All criteria met (Strongest Password).

Bottom Node (Ø): No criteria met (Weakest Password).

intermediate Nodes: Combinations of criteria (e.g.,A∩C,B∩D)

Example:



PASSWORD STRENGTH CHECKER AND RELATIONS

- Attribute Relations:
  - Length vs Entropy

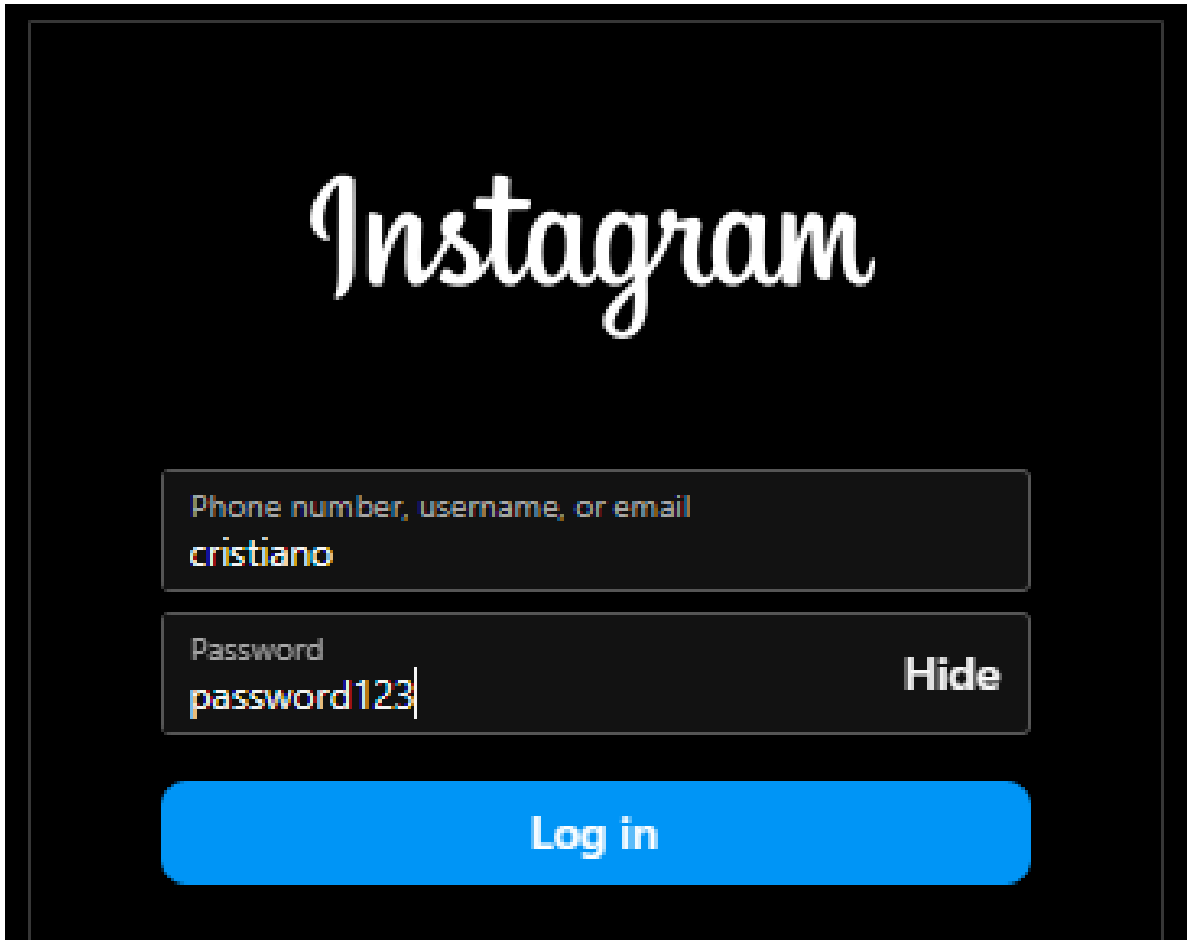
- Functions:
  f(password) = strength

- Inverse relation:
  f^-1(strong) = password

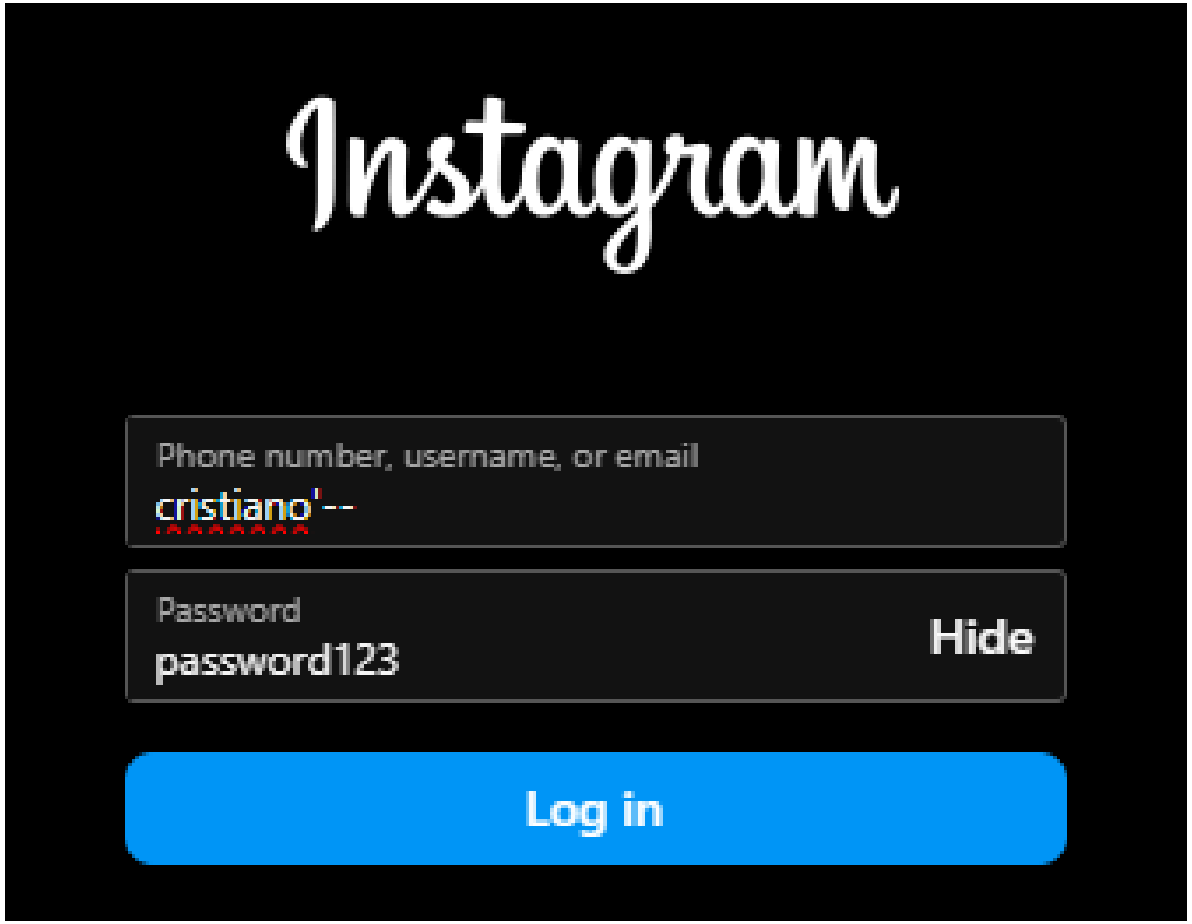
SQL INJECTION

SELECT * FROM username
WHERE username =
'cristiano'

AND password =
'password123'



SELECT * FROM users
WHERE username =
'cristiano'-- 'AND
password = 'password123'



PASSWORD STRENGTH CHECKER AND PROBABILITY

- Estimates time in which attacker can guess a password
  -Total number of possible password combinations

- Attack strategies:
  - Brute-Force attack
  - Dictionary-Based method

CALCULATING PROBABLITY:

- P = 1/N (N = c^L)

For example: For strong password (length = 6) that contains uppercase, lowercase, numbers and special characters the probability is:

c = 26+26+10+32=94

L = 6

N=94^6

P=1/94^6

PASSWORD STRENGTH CHECKER CODE

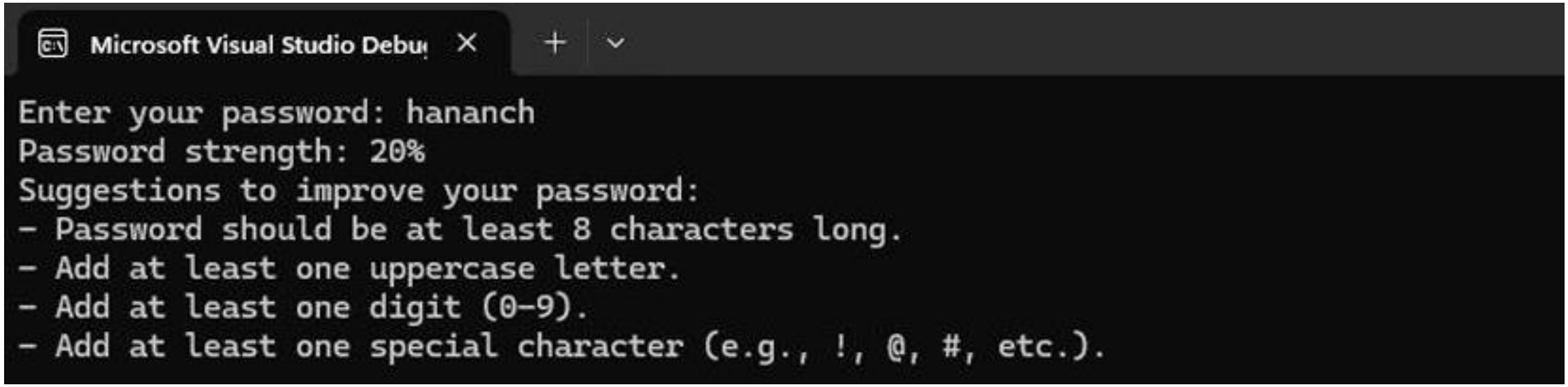
```
bool length(char password[]) {
  for (int i = 0; password[i] != '\0'; i++) {
      if (i > 7) return true;
   }
   return false;
}
bool isUppercase(char password[]) {
   for (int i = 0; password[i] != '\0'; i++)
   {
      if (password[i] > 66 && password[i] < 91)
         return true;
   }
   return false;
}
```

```
bool isLowercase(char password[]) {
    for (int i = 0; password[i] != '\0'; i++)
    {
        if (password[i] > 96 && password[i] < 123)
            return true;
    }
    return false;
}
bool isDigit(char password[]) {
    for (int i = 0; password[i] != '\0'; i++)
    {
        if (password[i] > 47 && password[i] < 58)
            return true;
    }
    return false;
}
```

```
bool isSpecialChar(char password[]) {
    for (int i = 0; password[i] != '\0'; i++)
    {
        if ((password[i] > 31 && password[i] < 48) ||
(password[i]           > 90 && password[i] < 97) || (password[i]
> 122 && password[i] <   127)) {
            return true;
      }
    }
    return false;
}
```

```
int PasswordStrength(char password[], string& feedback) {
    int score = 0;
    int totalCriteria = 5;
    if (length(password))
        score += 1;
    else
        feedback += "- Password should be at least 8 characters long.\n";
    if (isUppercase(password))
        score += 1;
    else
        feedback += "- Add at least one uppercase letter.\n";
    if (isLowercase(password))
        score += 1;
    else
        feedback += "- Add at least one lowercase letter.\n";
    if (isDigit(password))
        score += 1;
    else
        feedback += "- Add at least one digit (0-9).\n";
    if (isSpecialChar(password))
        score += 1;
    else
        feedback += "- Add at least one special character (e.g., !, @, #, etc.).\n";
    int percentage = (score * 100) / totalCriteria;
    return percentage;
}
```

```
int main() {
    char  password[100];
    string feedback = "";
    cout << "Enter your password: ";
    cin.getline(password, 100);
    int strengthPercentage = PasswordStrength(password,
feedback);
    cout << "Password strength: " << strengthPercentage << "%" <<
endl;
    if (strengthPercentage < 100)
        cout << "Suggestions to improve your password: " << endl
<< feedback;
    else
        cout << "Your password is strong!" << endl;
    return 0;
}
```


Microsoft Visual Studio Debu
Enter your password: hananch
Password strength: 20%
Suggestions to improve your password:
- Password should be at least 8 characters long.
- Add at least one uppercase letter.
- Add at least one digit (0-9).
- Add at least one special character (e.g., !, @, #, etc.).

REAL WORLD APPLICATIONS

1. Educational Platforms:

Student and Faculty

Online Learning Platforms

2. IoT (Internet of Things) Devices:

Automated home lockdown system

3. Cryptocurrency Platforms:

4. Personal Branding Websites:

Noteworthy historical incidents resulting from weak passwords

1. The 2014 Sony Pictures Hack:
   - Hackers gained access to Sony's network
   - Personal details of 6,000 employees , thousands of private emails

2. The 2020 Twitter Hack:
   - Hackers gained access to Twitter’s internal systems
   - Bitcoin scam

CONCLUSION

Password Strength Checker is not just a technical tool but a critical component of a comprehensive security strategy.