Jaypee Institute of Information Technology, Noida

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING AND INFORMATION TECHNOLOGY

Project Title: Cryptography and Password Security

Enroll. No.	**Name of Student**
9921103098	Satvik Buttan
9921103145	Rahi Agarwal
9921103191	Naman Jhanwar

Course Name: Information Security Lab
Course Code: 15B17CI576
Program: B. Tech. CS&E
3rd Year 5th Sem

2023 - 2024

Table of Contents

Introduction

In an era defined by digital interactions, the Cryptography and Password Security Mini Project offers a hands-on journey into classical encryption methods and vital password protection practices. This project unravels the art of encryption through Caesar, Xor, and Vigenère ciphers, while also emphasizing the critical role of strong passwords. Exploring password strength assessment, time estimation for cracking,and security features against repeated incorrect entries, this endeavor equips us with essential tools to navigate the evolving digital landscape. Join us to unravel the secrets of secure communication, fortify our digital presence, and champion data protection in the modern age.

Problem Statement

In today's digital age, data security is of paramount importance. This mini project aims to explore the world of classical encryption techniques and password security measures. The project involves implementing three classical ciphers – Caesar Cipher, Xor Cipher, and Vigenère Cipher – and assessing the strength of user-generated passwords. Additionally, the project seeks to estimate the time it takes to crack passwords and implement a security feature that engages if repeated incorrect passwords are entered.

Motivation

"In today's interconnected world, where data privacy and security are paramount, the Cryptography and Password Security Mini Project emerges as a beacon of knowledge and empowerment. With the digital realm becoming increasingly integral to our lives, understanding the fundamentals of cryptography and robust password protection is no longer a choice but a necessity.

This project invites you on an enlightening journey through the annals of encryption techniques. From the time-tested Caesar, Xor, and Vigenère ciphers to the latest advances in password security, we delve into the intricate world of safeguarding digital information.

Our mission is twofold. Firstly, we aim to demystify the science of encryption, making it accessible and understandable to all. Through hands-on exploration, you'll gain insights into the inner workings of these classical ciphers and, more importantly, how they continue to underpin modern security practices.

Secondly, we delve into the critical realm of password protection, which serves as the first line of defense against digital threats. With a focus on assessing password strength, estimating the time required for unauthorized access, and implementing safeguards against brute-force attacks, we equip you with the skills and knowledge needed to secure your digital presence.

1.1 Objective

- **Cryptography Implementation:**

Implement the encryption and decryption functions for Caesar Cipher, Xor Cipher, and Vigenère Cipher .Provide users the ability to encrypt and decrypt messages using these ciphers.

- **Password Strength Assessmen**t:

Develop a password strength evaluation mechanism based on specified criteria.

Guide users to create strong passwords by enforcing recommended practices.

- **Password Cracking Time Estimation:**

Devise an algorithm to estimate the time it would take to crack a given password using brute-force techniques. Use computational power estimation and password complexity factors for accurate predictions.

- **Security Feature Implementation:**

Integrate a security measure that activates if a user enters the wrong password multiple times. Upon exceeding a threshold, initiate a "monitor sleep" function that puts the monitor into sleep mode for a predefined duration

1.2 Detailed description and Implementation of the project

The project aims to provide a comprehensive information security system that combines multiple encryption methods and includes utilities for password generation and checking. The code is organized into different sections, each serving a unique purpose.

- **User Authentication:**

  The project starts with user authentication through a password. Users need to enter the correct password to access the information security system.

  - The program starts by asking the user for a password.

  Users must enter the correct password to access the system. If the password is incorrect, the program allows a limited number of attempts before taking action

- **Menu-Based Interface:**

  Once authenticated, users are presented with a menu-based interface to choose from various security and utility features.

- **Caesar Cipher:**

  Users can select the Caesar Cipher option, where they can input plaintext and a key to encrypt and decrypt messages using the Caesar Cipher algorithm. The Caesar Cipher provides a simple letter-shifting technique for data encryption.

  - If the user selects the Caesar Cipher option:
  - They input a plaintext message.
  - They input an integer key for encryption.
  - The program encrypts the message using the Caesar Cipher.
  - The encrypted message is displayed.

  The program can also decrypt the message with the same key if desired

- **Xor Cipher:**

  This encryption technique works by applying the exclusive OR (XOR) operation between each character of the plaintext and the corresponding character in the key. This process encrypts and decrypts data, ensuring that the same key used for encryption is also used for decryption.

- If the user selects the Vernam Cipher option:

-They input a plaintext message.

- They input a secret key generated as a sequence of random numbers.

- The program encrypts the message using the XOR cipher (simulating Vernam Cipher) with the generated random key.

- The encrypted message is displayed.

- To decrypt the message, the user needs to input the same key used for encryption.

-The program decrypts the message using the XOR cipher, retrieving the original plaintext.

- **Vigenere Cipher:**
  Users can choose the Vigenere Cipher option, providing plaintext and a keyword for encryption and decryption. The Vigenere Cipher employs a polyalphabetic substitution technique, making it more secure than simple substitution ciphers.

  - If the user selects the Vigenere Cipher option:
  - They input a plaintext message.
  - They input a keyword consisting of alphabetic characters.
  - The program encrypts the message using the Vigenere Cipher.
  - The encrypted message is displayed.
  - The program can also decrypt the message with the same keyword if desired.

- **Password Generator:**
  The system includes a password generator utility. Users can create strong random passwords with a combination of lowercase letters, uppercase letters, numbers, and special characters. A generated password is displayed for the user.

  - If the user selects the Password Generator option:
  - The program generates a random strong password that includes lowercase letters, uppercase letters, numbers, and special characters.
  - The generated password is displayed for the user.

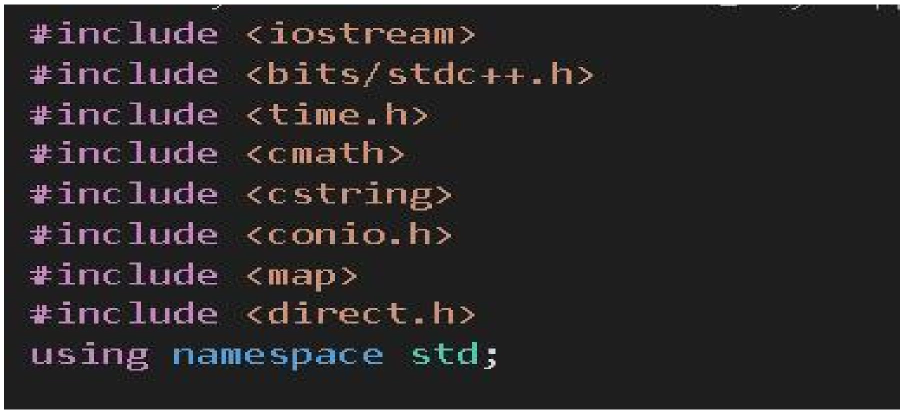
- **Password Checker:**

  The password checker utility allows users to input a password, and the system assesses its strength. The criteria for a strong password include length, the presence of uppercase letters, special characters, and numbers. The system provides feedback on whether the password is strong or weak.

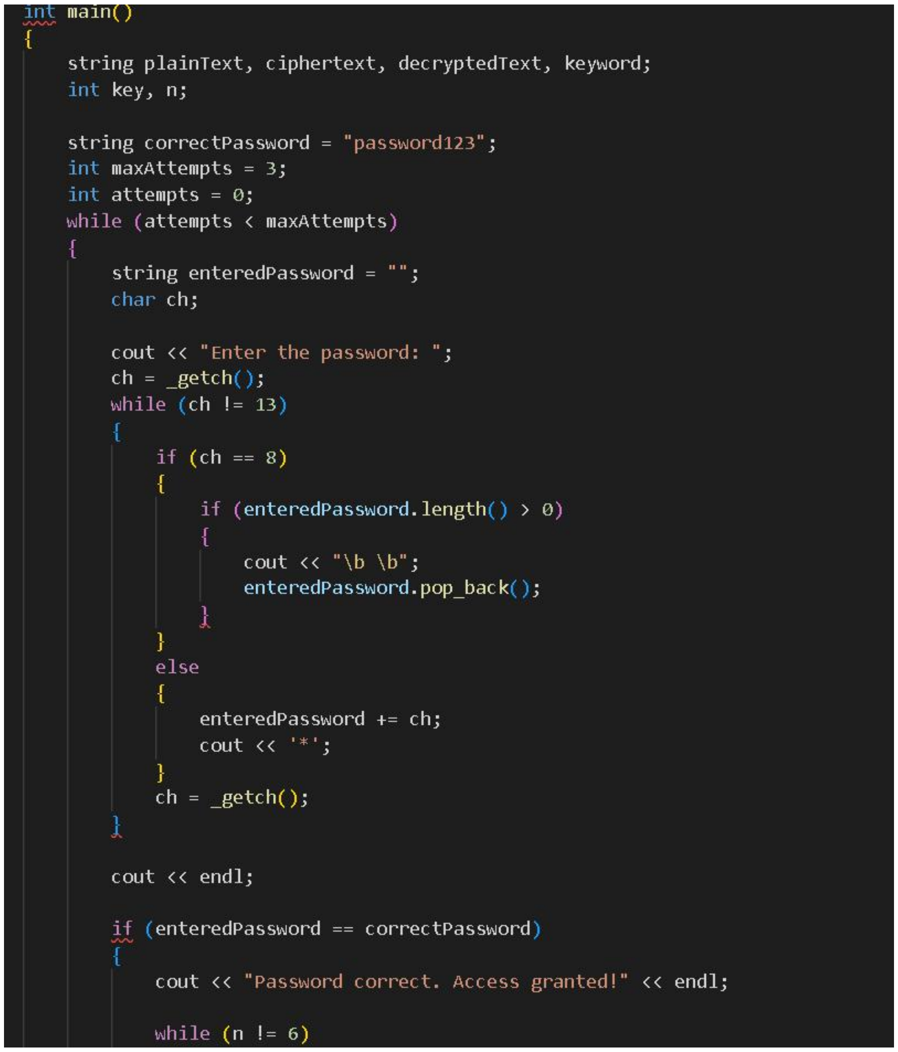
  - If the user selects the Password Checker option:
  - The user inputs a password.
  - The program checks the password for strength based on predefined criteria (length, uppercase letters, special characters, and numbers).
  - The program provides feedback on whether the password is strong or weak.

2.1 Program Code

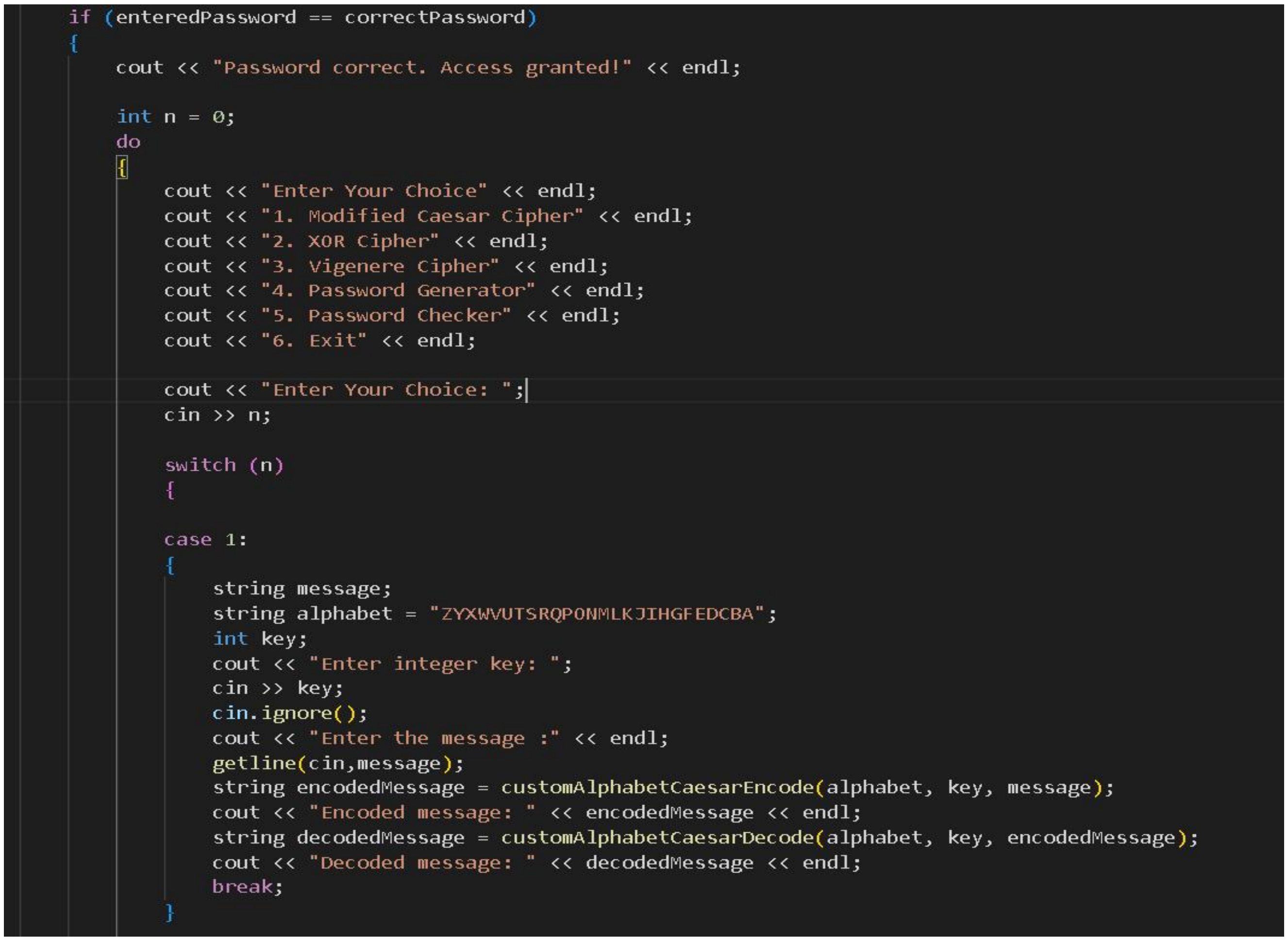
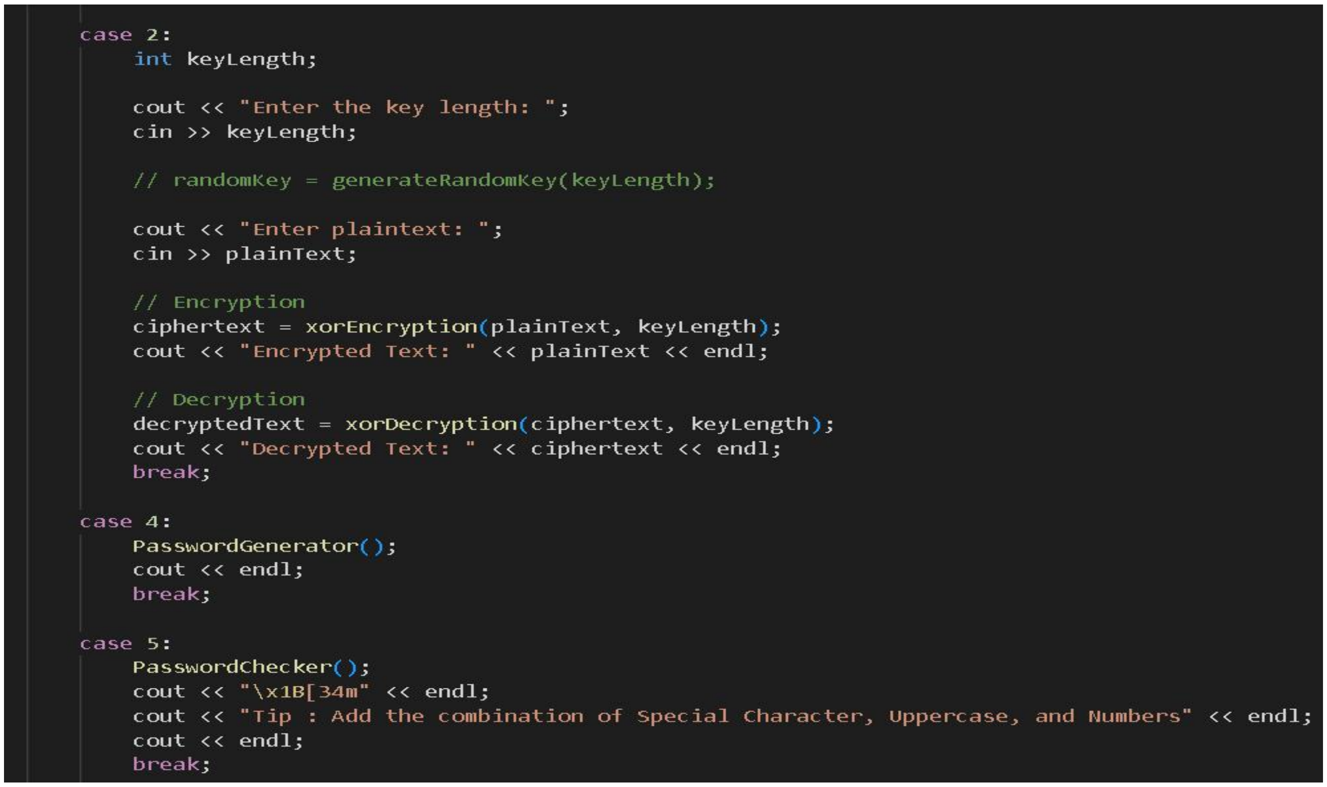
Main function

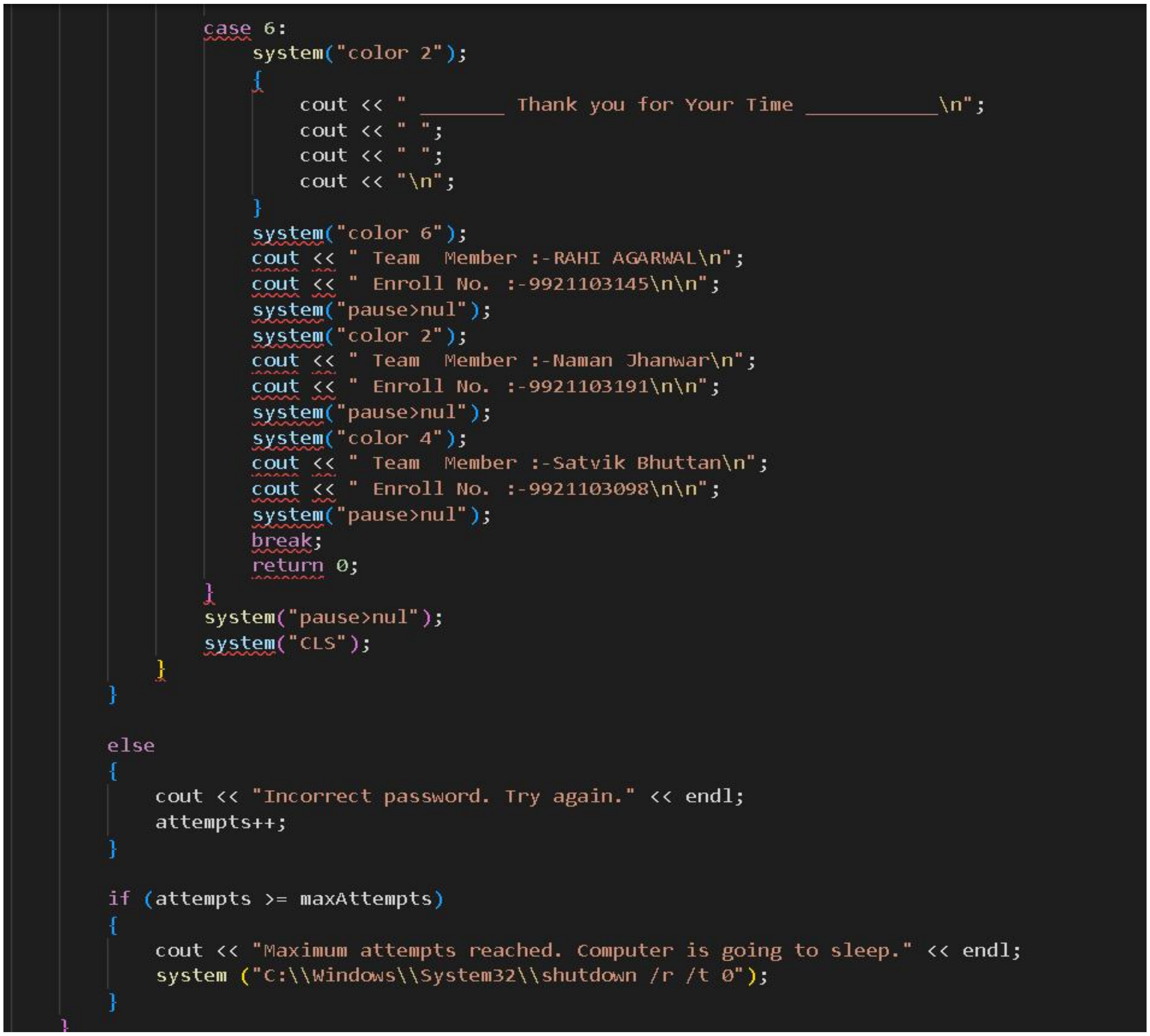


```
#include <iostream>
#include <bits/stdc++.h>
#include <time.h>
#include <cmath>
#include <cstring>
#include <conio.h>
#include <map>
#include <direct.h>
using namespace std;
```

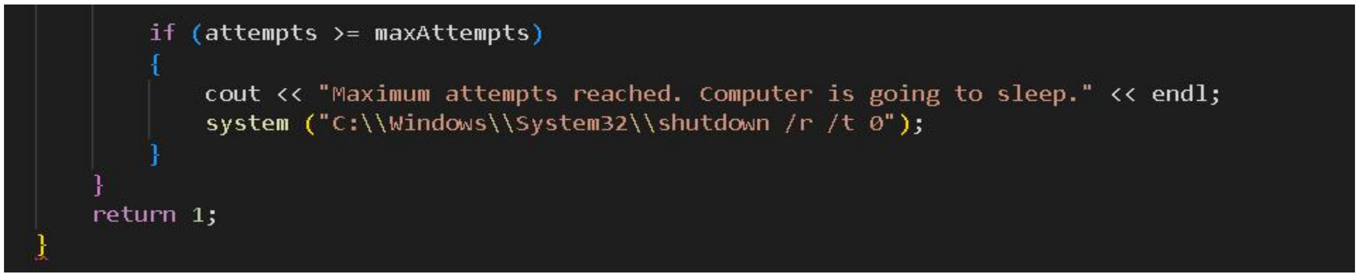


```
int main()
{
    string plainText, ciphertext, decryptedText, keyword;
    int key, n;

    string correctPassword = "password123";
    int maxAttempts = 3;
    int attempts = 0;
    while (attempts < maxAttempts)
    {
        string enteredPassword = "";
        char ch;

        cout << "Enter the password: ";
        ch = _getch();
        while (ch != 13)
        {
            if (ch == 8)
            {
                if (enteredPassword.length() > 0)
                {
                    cout << "\b \b";
                    enteredPassword.pop_back();
                }
            }
            else
            {
                enteredPassword += ch;
                cout << '*';
            }
            ch = _getch();
        }

        cout << endl;

        if (enteredPassword == correctPassword)
        {
            cout << "Password correct. Access granted!" << endl;

            while (n != 6)
```


```cpp
if (enteredPassword == correctPassword)
{
    cout << "Password correct. Access granted!" << endl;

    int n = 0;
    do
    {
        cout << "Enter Your Choice" << endl;
        cout << "1. Modified Caesar Cipher" << endl;
        cout << "2. XOR Cipher" << endl;
        cout << "3. Vigenere Cipher" << endl;
        cout << "4. Password Generator" << endl;
        cout << "5. Password Checker" << endl;
        cout << "6. Exit" << endl;

        cout << "Enter Your Choice: ";
        cin >> n;

        switch (n)
        {

        case 1:
        {
            string message;
            string alphabet = "ZYXWVUTSRQPONMLKJIHGFEDCBA";
            int key;
            cout << "Enter integer key: ";
            cin >> key;
            cin.ignore();
            cout << "Enter the message :" << endl;
            getline(cin,message);
            string encodedMessage = customAlphabetCaesarEncode(alphabet, key, message);
            cout << "Encoded message: " << encodedMessage << endl;
            string decodedMessage = customAlphabetCaesarDecode(alphabet, key, encodedMessage);
            cout << "Decoded message: " << decodedMessage << endl;
            break;
        }
```

```cpp
case 2:
    int keyLength;

    cout << "Enter the key length: ";
    cin >> keyLength;

    // randomKey = generateRandomKey(keyLength);

    cout << "Enter plaintext: ";
    cin >> plainText;

    // Encryption
    ciphertext = xorEncryption(plainText, keyLength);
    cout << "Encrypted Text: " << plainText << endl;

    // Decryption
    decryptedText = xorDecryption(ciphertext, keyLength);
    cout << "Decrypted Text: " << ciphertext << endl;
    break;

case 4:
    PasswordGenerator();
    cout << endl;
    break;

case 5:
    PasswordChecker();
    cout << "\x1B[34m" << endl;
    cout << "Tip : Add the combination of Special Character, Uppercase, and Numbers" << endl;
    cout << endl;
    break;
```

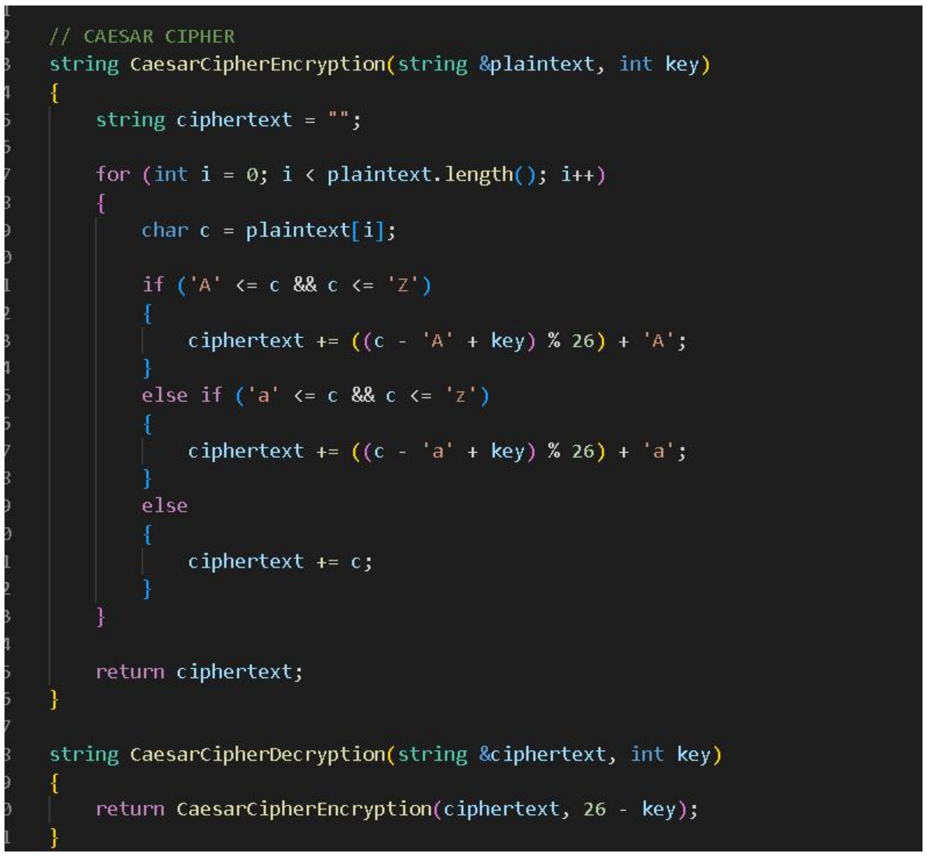


```
            case 6:
                system("color 2");
                {
                    cout << " _______ Thank you for Your Time ___________\n";
                    cout << " ";
                    cout << " ";
                    cout << "\n";
                }
                system("color 6");
                cout << " Team  Member :-RAHI AGARWAL\n";
                cout << " Enroll No. :-9921103145\n\n";
                system("pause>nul");
                system("color 2");
                cout << " Team  Member :-Naman Jhanwar\n";
                cout << " Enroll No. :-9921103191\n\n";
                system("pause>nul");
                system("color 4");
                cout << " Team  Member :-Satvik Bhuttan\n";
                cout << " Enroll No. :-9921103098\n\n";
                system("pause>nul");
                break;
                return 0;
            }
            system("pause>nul");
            system("CLS");
        }
    }

    else
    {
        cout << "Incorrect password. Try again." << endl;
        attempts++;
    }

    if (attempts >= maxAttempts)
    {
        cout << "Maximum attempts reached. Computer is going to sleep." << endl;
        system ("C:\\Windows\\System32\\shutdown /r /t 0");
    }
}
```



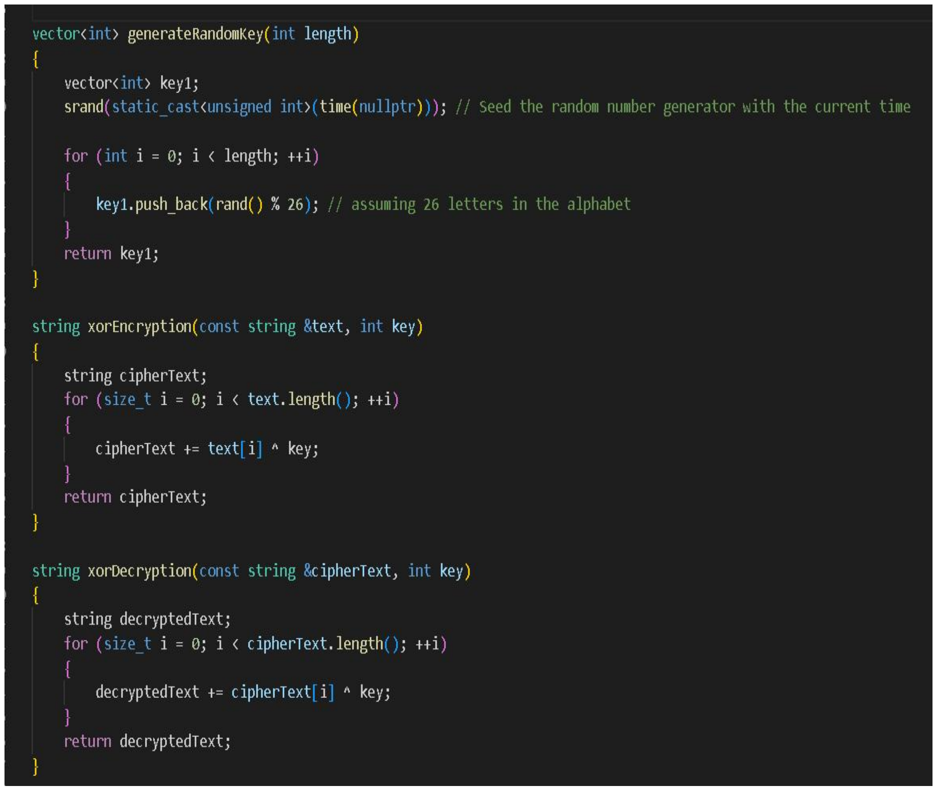
```
        if (attempts >= maxAttempts)
        {
            cout << "Maximum attempts reached. Computer is going to sleep." << endl;
            system ("C:\\Windows\\System32\\shutdown /r /t 0");
        }
    }
    return 1;
}
```

Caesar Cipher



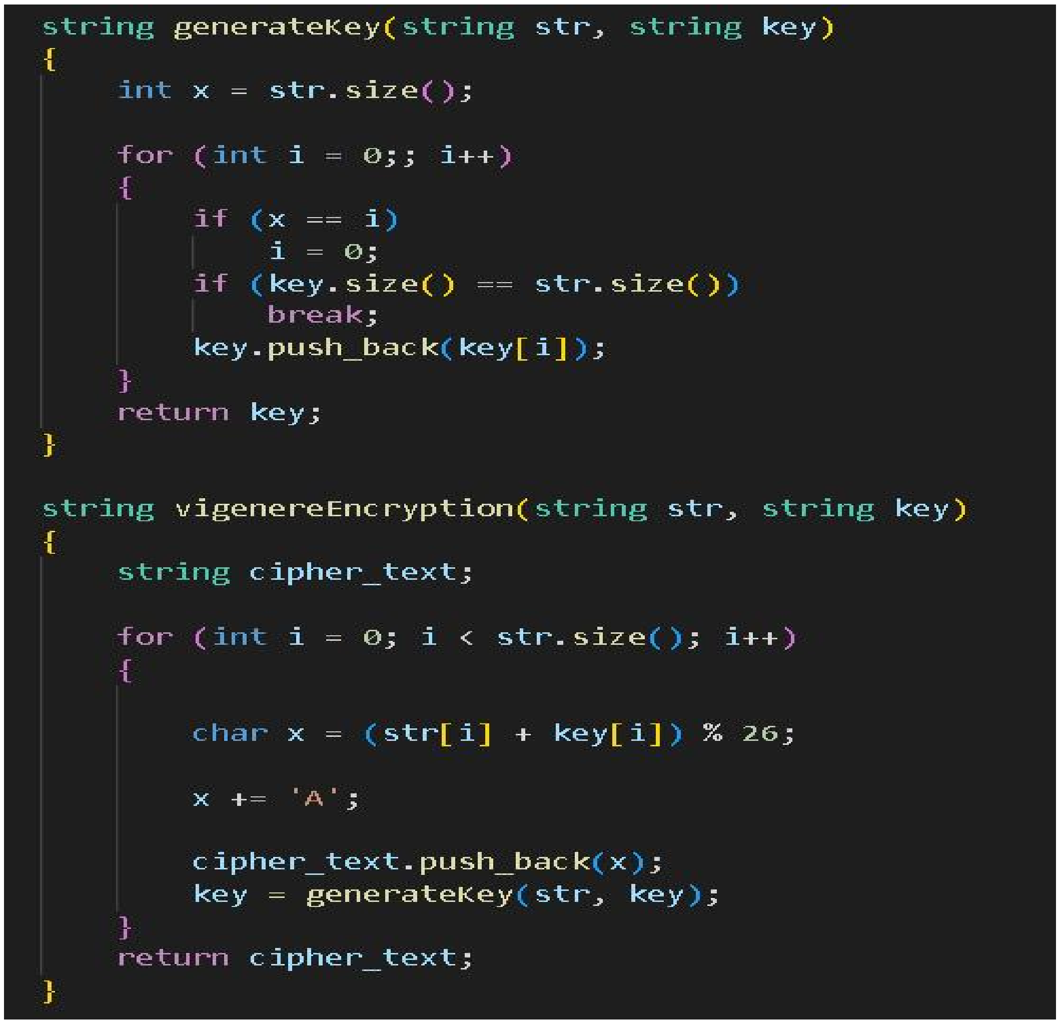
```cpp
// CAESAR CIPHER
string CaesarCipherEncryption(string &plaintext, int key)
{
    string ciphertext = "";

    for (int i = 0; i < plaintext.length(); i++)
    {
        char c = plaintext[i];

        if ('A' <= c && c <= 'Z')
        {
            ciphertext += ((c - 'A' + key) % 26) + 'A';
        }
        else if ('a' <= c && c <= 'z')
        {
            ciphertext += ((c - 'a' + key) % 26) + 'a';
        }
        else
        {
            ciphertext += c;
        }
    }

    return ciphertext;
}

string CaesarCipherDecryption(string &ciphertext, int key)
{
    return CaesarCipherEncryption(ciphertext, 26 - key);
}
```

XOR Cipher

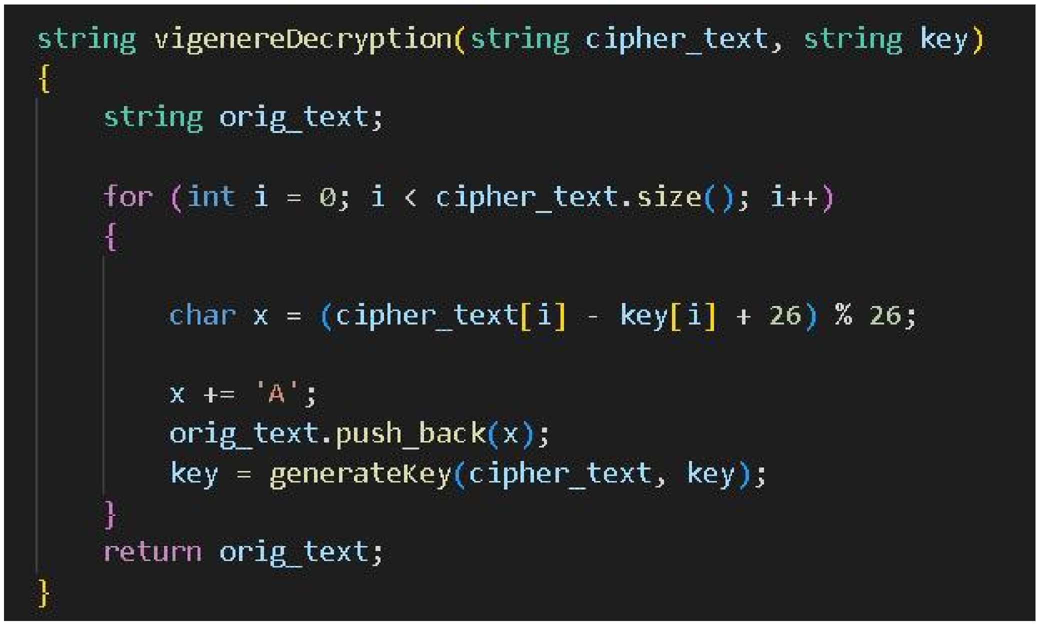


```cpp
vector<int> generateRandomKey(int length)
{
    vector<int> key1;
    srand(static_cast<unsigned int>(time(nullptr))); // Seed the random number generator with the current time

    for (int i = 0; i < length; ++i)
    {
        key1.push_back(rand() % 26); // assuming 26 letters in the alphabet
    }
    return key1;
}

string xorEncryption(const string &text, int key)
{
    string cipherText;
    for (size_t i = 0; i < text.length(); ++i)
    {
        cipherText += text[i] ^ key;
    }
    return cipherText;
}

string xorDecryption(const string &cipherText, int key)
{
    string decryptedText;
    for (size_t i = 0; i < cipherText.length(); ++i)
    {
        decryptedText += cipherText[i] ^ key;
    }
    return decryptedText;
}
```

Viegener Cipher

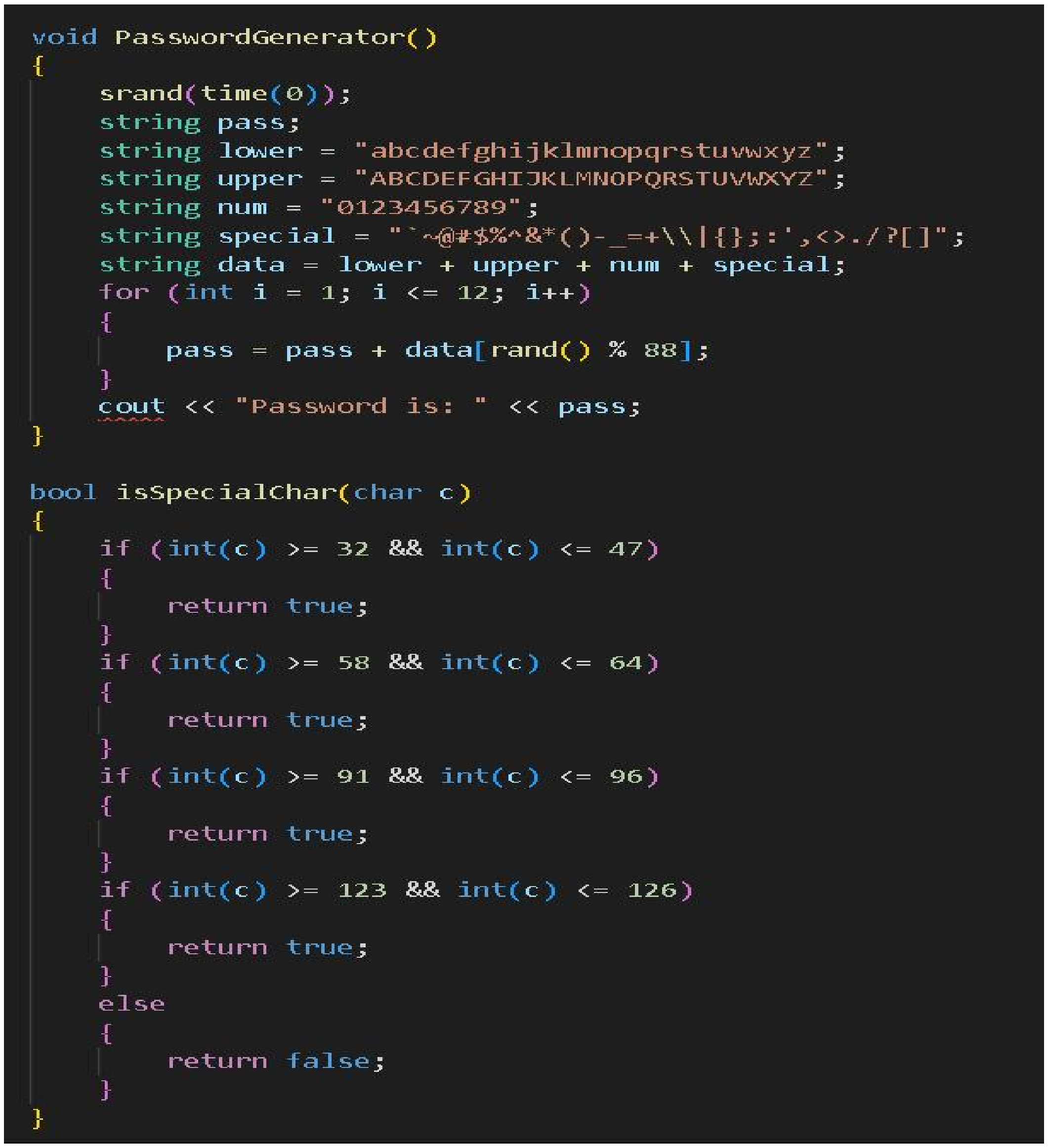


```
string generateKey(string str, string key)
{
    int x = str.size();

    for (int i = 0;; i++)
    {
        if (x == i)
            i = 0;
        if (key.size() == str.size())
            break;
        key.push_back(key[i]);
    }
    return key;
}

string vigenereEncryption(string str, string key)
{
    string cipher_text;

    for (int i = 0; i < str.size(); i++)
    {

        char x = (str[i] + key[i]) % 26;

        x += 'A';

        cipher_text.push_back(x);
        key = generateKey(str, key);
    }
    return cipher_text;
}
```



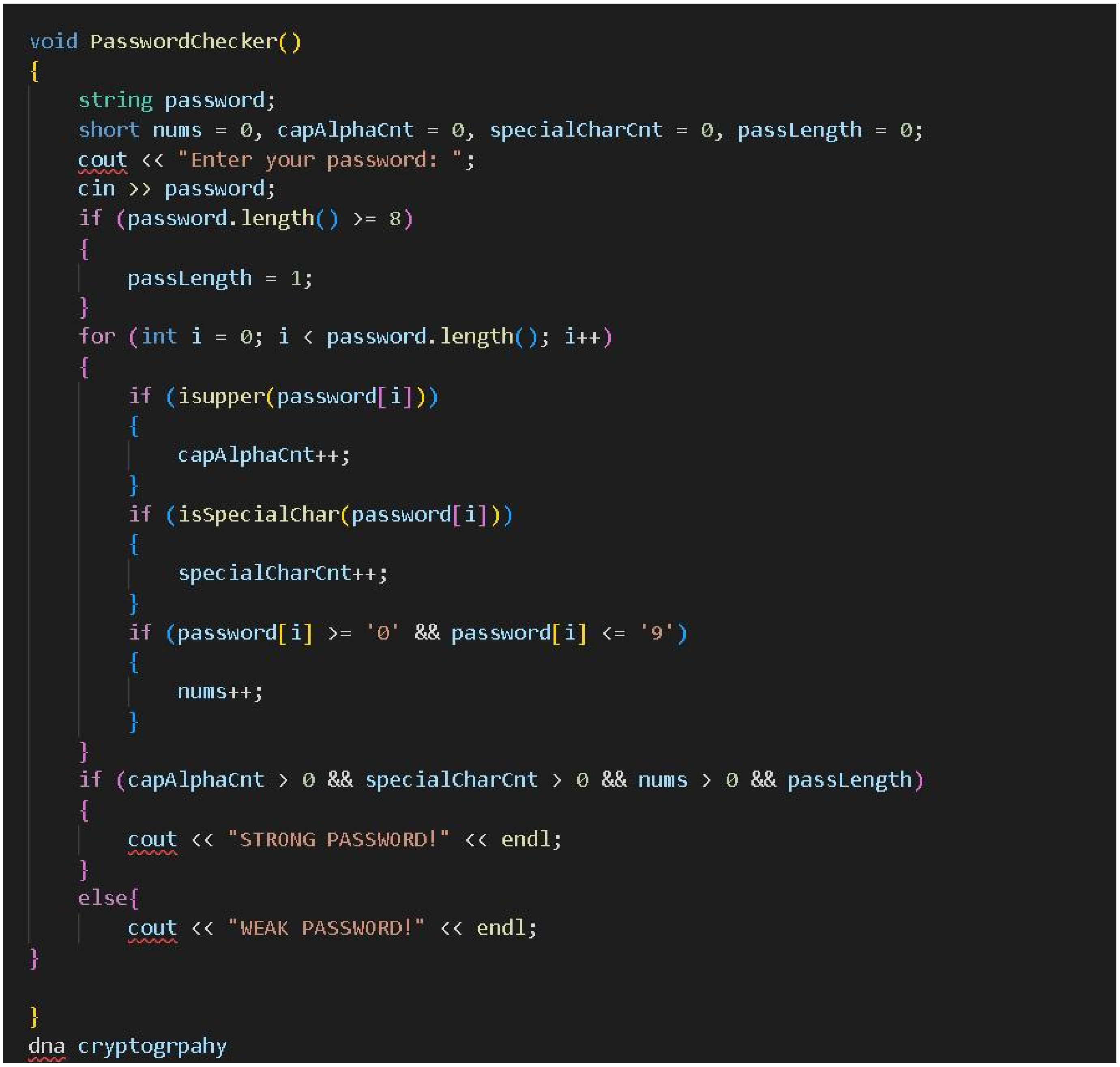
```
string vigenereDecryption(string cipher_text, string key)
{
    string orig_text;

    for (int i = 0; i < cipher_text.size(); i++)
    {

        char x = (cipher_text[i] - key[i] + 26) % 26;

        x += 'A';
        orig_text.push_back(x);
        key = generateKey(cipher_text, key);
    }
    return orig_text;
}
```

Password Generator



```cpp
void PasswordGenerator()
{
    srand(time(0));
    string pass;
    string lower = "abcdefghijklmnopqrstuvwxyz";
    string upper = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    string num = "0123456789";
    string special = "`~@#$%^&*()-_=+\\|{};:',<>./?[]";
    string data = lower + upper + num + special;
    for (int i = 1; i <= 12; i++)
    {
        pass = pass + data[rand() % 88];
    }
    cout << "Password is: " << pass;
}

bool isSpecialChar(char c)
{
    if (int(c) >= 32 && int(c) <= 47)
    {
        return true;
    }
    if (int(c) >= 58 && int(c) <= 64)
    {
        return true;
    }
    if (int(c) >= 91 && int(c) <= 96)
    {
        return true;
    }
    if (int(c) >= 123 && int(c) <= 126)
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

Password Checker

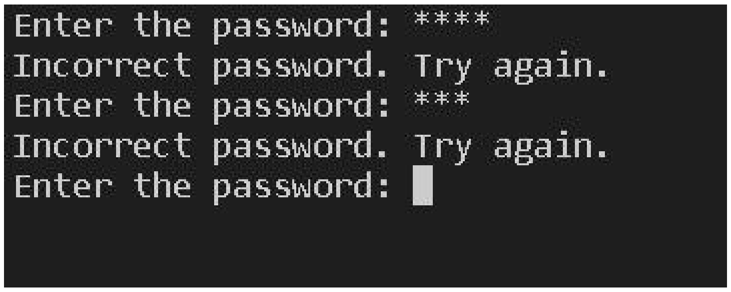


```
void PasswordChecker()
{
    string password;
    short nums = 0, capAlphaCnt = 0, specialCharCnt = 0, passLength = 0;
    cout << "Enter your password: ";
    cin >> password;
    if (password.length() >= 8)
    {
        passLength = 1;
    }
    for (int i = 0; i < password.length(); i++)
    {
        if (isupper(password[i]))
        {
            capAlphaCnt++;
        }
        if (isSpecialChar(password[i]))
        {
            specialCharCnt++;
        }
        if (password[i] >= '0' && password[i] <= '9')
        {
            nums++;
        }
    }
    if (capAlphaCnt > 0 && specialCharCnt > 0 && nums > 0 && passLength)
    {
        cout << "STRONG PASSWORD!" << endl;
    }
    else{
        cout << "WEAK PASSWORD!" << endl;
}

}
dna cryptogrpahy
```

3.1 Results

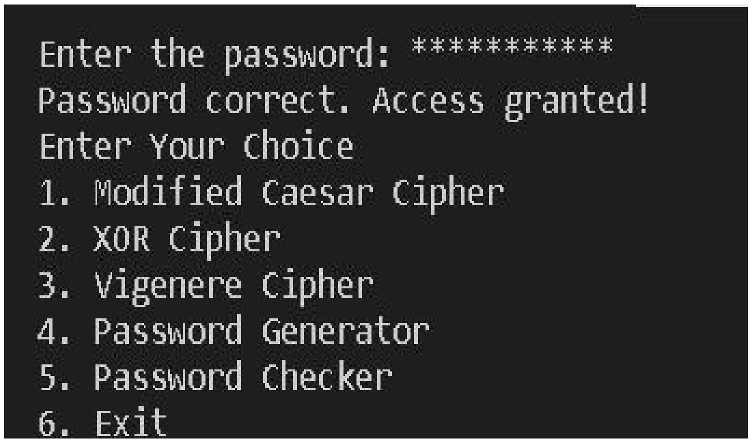
First of all, we have to give our set password to run the code.

If we give more than 2 incorrect try , then our system will restart automatically.



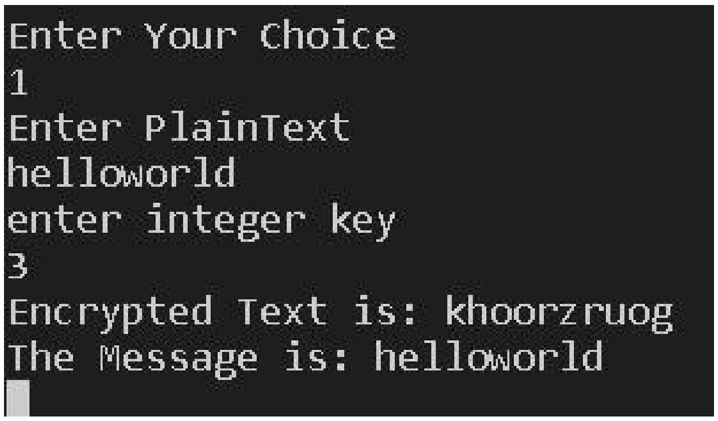
```
Enter the password: ****
Incorrect password. Try again.
Enter the password: ***
Incorrect password. Try again.
Enter the password: 
```

In above figure, if we type incorrect password again our system will restart automatically.



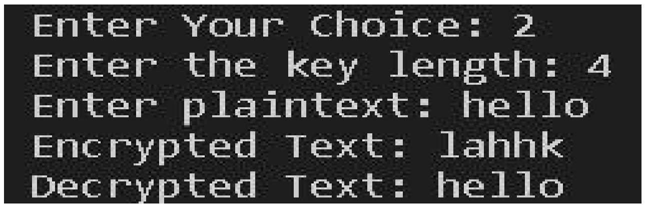
```
Enter the password: ***********
Password correct. Access granted!
Enter Your Choice
1. Modified Caesar Cipher
2. XOR Cipher
3. Vigenere Cipher
4. Password Generator
5. Password Checker
6. Exit
```

Caesar Cipher



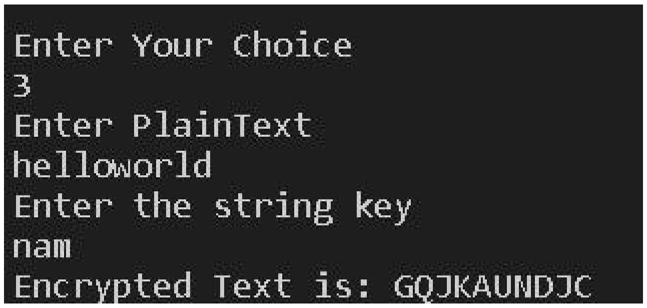
```
Enter Your Choice
1
Enter PlainText
helloworld
enter integer key
3
Encrypted Text is: khoorzruog
The Message is: helloworld
```

XOR Cipher:



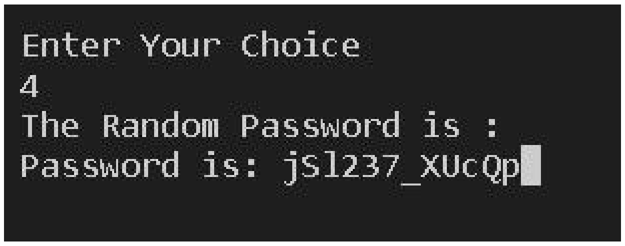
```
Enter Your Choice: 2
Enter the key length: 4
Enter plaintext: hello
Encrypted Text: lahhk
Decrypted Text: hello
```

Viegener Cipher:



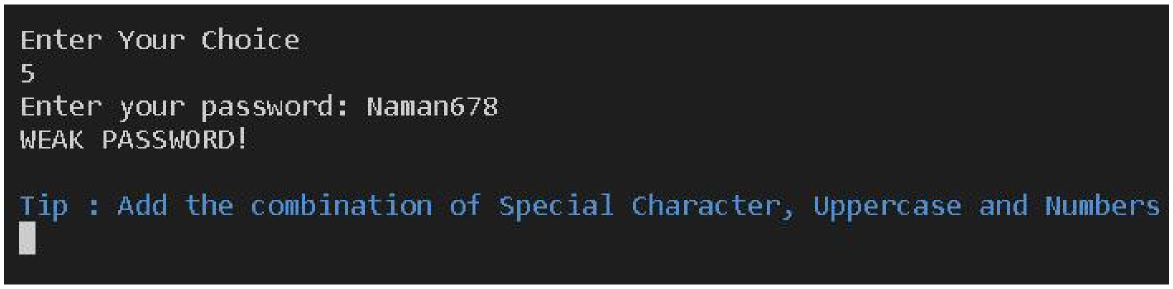
```
Enter Your Choice
3
Enter PlainText
helloworld
Enter the string key
nam
Encrypted Text is: GQJKAUNDJC
```

Password Generator:

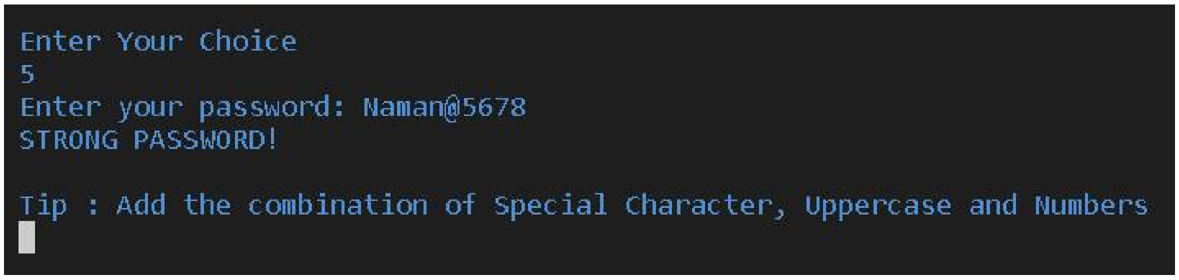


```
Enter Your Choice
4
The Random Password is :
Password is: jSl237_XUcQp
```

Password Checker:



```
Enter Your Choice
5
Enter your password: Naman678
WEAK PASSWORD!

Tip : Add the combination of Special Character, Uppercase and Numbers
```



```
Enter Your Choice
5
Enter your password: Naman@5678
STRONG PASSWORD!

Tip : Add the combination of Special Character, Uppercase and Numbers
```

Conclusion

In summary, our project provides comprehensive solution for enhancing data security and improving password management. It incorporates user authentication, cryptography functions, and user-friendly features. By offering encryption capabilities and encouraging strong password practices, it serves as a valuable tool for enhancing information security. Additionally, the project recognizes and acknowledges the contributions of the team members, fostering a sense of teamwork and collaboration.

References

https://www.geeksforgeeks.org/caesar-cipher-in-cryptography/

https://www.javatpoint.com/vigenere-cipher

https://www.javatpoint.com/password-validation-in-cpp

https://www.includehelp.com/cpp-programs/generate-random-password.aspx