**Shift Ciphers**

**def additive\_cipher\_encrypt(message, key):**

**encrypted\_message = ""**

**for c in message:**

**if c.isalpha():**

**if c.isupper():**

**base = 'A'**

**else:**

**base = 'a'**

**shifted\_c = chr((ord(c) - ord(base) + key) % 26 + ord(base))**

**encrypted\_message += shifted\_c**

**else:**

**encrypted\_message += c**

**return encrypted\_message**

**def additive\_cipher\_decrypt(message, key):**

**decrypted\_message = ""**

**for c in message:**

**if c.isalpha():**

**if c.isupper():**

**base = 'A'**

**else:**

**base = 'a'**

**shifted\_c = chr((ord(c) - ord(base) - key) % 26 + ord(base))**

**decrypted\_message += shifted\_c**

**else:**

**decrypted\_message += c**

**return decrypted\_message**

**def multiplicative\_cipher\_encrypt(message, key):**

**encrypted\_message = ""**

**for c in message:**

**if c.isalpha():**

**if c.isupper():**

**base = 'A'**

**else:**

**base = 'a'**

**shifted\_c = chr((ord(c) - ord(base)) \* key % 26 + ord(base))**

**encrypted\_message += shifted\_c**

**else:**

**encrypted\_message += c**

**return encrypted\_message**

**def multiplicative\_cipher\_decrypt(message, key):**

**try:**

**inverse = pow(key, -1, 26)**

**except ValueError:**

**return print("Key is not invertible!")**

**decrypted\_message = ""**

**for c in message:**

**if c.isalpha():**

**if c.isupper():**

**base = 'A'**

**else:**

**base = 'a'**

**shifted\_c = chr((ord(c) - ord(base)) \* inverse % 26 + ord(base))**

**decrypted\_message += shifted\_c**

**else:**

**decrypted\_message += c**

**return decrypted\_message**

**def affine\_cipher\_encrypt(message, a, b):**

**encrypted\_message = ""**

**for c in message:**

**if c.isalpha():**

**if c.isupper():**

**base = 'A'**

**else:**

**base = 'a'**

**shifted\_c = chr(((ord(c) - ord(base)) \* a + b) % 26 + ord(base))**

**encrypted\_message += shifted\_c**

**else:**

**encrypted\_message += c**

**return encrypted\_message**

**def affine\_cipher\_decrypt(message, a, b):**

**try:**

**inverse\_a = pow(a, -1, 26)**

**except ValueError:**

**return print("Key is not invertible!")**

**decrypted\_message = ""**

**for c in message:**

**if c.isalpha():**

**if c.isupper():**

**base = 'A'**

**else:**

**base = 'a'**

**shifted\_c = chr(((ord(c) - ord(base) - b) \* inverse\_a) % 26 + ord(base))**

**decrypted\_message += shifted\_c**

**else:**

**decrypted\_message += c**

**return decrypted\_message**

**def main():**

**while True:**

**print("Menu:")**

**print("1. Additive Cipher Encryption")**

**print("2. Additive Cipher Decryption")**

**print("3. Multiplicative Cipher Encryption")**

**print("4. Multiplicative Cipher Decryption")**

**print("5. Affine Cipher Encryption")**

**print("6. Affine Cipher Decryption")**

**print("7. Exit")**

**choice = int(input("Enter your choice: "))**

**if choice == 1:**

**message = input("Enter the message to encrypt: ")**

**key = int(input("Enter the key: "))**

**print("Encrypted message: ", additive\_cipher\_encrypt(message, key))**

**elif choice == 2:**

**message = input("Enter the message to encrypt: ")**

**key = int(input("Enter the key: "))**

**print("Decrypted message: ", additive\_cipher\_decrypt(message, key))**

**elif choice == 3:**

**message = input("Enter the message to encrypt: ")**

**key = int(input("Enter the key: "))**

**print("Encrypted message: ", multiplicative\_cipher\_encrypt(message, key))**

**elif choice == 4:**

**message = input("Enter the message to encrypt: ")**

**key = int(input("Enter the key: "))**

**print("Decrypted message: ", multiplicative\_cipher\_decrypt(message, key))**

**elif choice == 5:**

**message = input("Enter the message to encrypt: ")**

**a = int(input("Enter the multiplier: "))**

**b = int(input("Enter the additive: "))**

**print("Encrypted message: ", affine\_cipher\_encrypt(message, a, b))**

**elif choice == 6:**

**message = input("Enter the message to encrypt: ")**

**a = int(input("Enter the multiplier: "))**

**b = int(input("Enter the additive: "))**

**print("Decrypted message: ", affine\_cipher\_decrypt(message, a, b))**

**elif choice == 7:**

**print("Exiting the program!")**

**break**

**else:**

**print("Invalid choice. Please choose a valid option.")**

**if \_\_name\_\_ == "\_\_main\_\_":**

**main()**

**Secure Login System using Shift Ciphers (Additive)**

**def shift\_cipher\_encrypt(message, key):**

**encrypted\_message = ""**

**for c in message:**

**if c.isalpha():**

**if c.isupper():**

**base = 'A'**

**else:**

**base = 'a'**

**shifted\_c = chr((ord(c) - ord(base) + key) % 26 + ord(base))**

**encrypted\_message += shifted\_c**

**else:**

**encrypted\_message += c**

**return encrypted\_message**

**def shift\_cipher\_decrypt(message, key):**

**decrypted\_message = ""**

**for c in message:**

**if c.isalpha():**

**if c.isupper():**

**base = 'A'**

**else:**

**base = 'a'**

**shifted\_c = chr((ord(c) - ord(base) - key) % 26 + ord(base))**

**decrypted\_message += shifted\_c**

**else:**

**decrypted\_message += c**

**return decrypted\_message**

**stored\_users = {}**

**def register\_user():**

**username = input("Enter a username: ")**

**if username in stored\_users:**

**print("Username already exists!")**

**return**

**password = input("Enter a password: ")**

**key = int(input("Enter the shift key for encryption: "))**

**encrypted\_password = shift\_cipher\_encrypt(password, key)**

**stored\_users[username] = (encrypted\_password, key)**

**print("User registered successfully!")**

**def login\_user():**

**username = input("Enter your username: ")**

**if username not in stored\_users:**

**print("User does not exist!")**

**return**

**password = input("Enter your password: ")**

**encrypted\_password, key = stored\_users[username]**

**if encrypted\_password == shift\_cipher\_encrypt(password, key):**

**print("Login successful!")**

**else:**

**print("Incorrect password!")**

**def main():**

**while True:**

**print("\nMenu:")**

**print("1. Register")**

**print("2. Login")**

**print("3. Exit")**

**choice = int(input("Enter your choice: "))**

**if choice == 1:**

**register\_user()**

**elif choice == 2:**

**login\_user()**

**elif choice == 3:**

**print("Exiting!")**

**break**

**else:**

**print("Invalid choice. Please choose a valid option.")**

**if \_\_name\_\_ == "\_\_main\_\_":**

**main()**

**Secure Login System using Shift Ciphers (Multiplicative)**

**def shift\_cipher\_encrypt(message, key):**

**encrypted\_message = ""**

**for c in message:**

**if c.isalpha():**

**if c.isupper():**

**base = 'A'**

**else:**

**base = 'a'**

**shifted\_c = chr((ord(c) - ord(base)) \* key % 26 + ord(base))**

**encrypted\_message += shifted\_c**

**else:**

**encrypted\_message += c**

**return encrypted\_message**

**def shift\_cipher\_decrypt(message, key):**

**try:**

**inverse = pow(key, -1, 26)**

**except ValueError:**

**return print("Key is not invertible!")**

**decrypted\_message = ""**

**for c in message:**

**if c.isalpha():**

**if c.isupper():**

**base = 'A'**

**else:**

**base = 'a'**

**shifted\_c = chr((ord(c) - ord(base)) \* inverse % 26 + ord(base))**

**decrypted\_message += shifted\_c**

**else:**

**decrypted\_message += c**

**return decrypted\_message**

**stored\_users = {}**

**def register\_user():**

**username = input("Enter a username: ")**

**if username in stored\_users:**

**print("Username already exists!")**

**return**

**password = input("Enter a password: ")**

**key = int(input("Enter the shift key for encryption: "))**

**encrypted\_password = shift\_cipher\_encrypt(password, key)**

**stored\_users[username] = (encrypted\_password, key)**

**print("User registered successfully!")**

**def login\_user():**

**username = input("Enter your username: ")**

**if username not in stored\_users:**

**print("User does not exist!")**

**return**

**password = input("Enter your password: ")**

**encrypted\_password, key = stored\_users[username]**

**if encrypted\_password == shift\_cipher\_encrypt(password, key):**

**print("Login successful!")**

**else:**

**print("Incorrect password!")**

**def main():**

**while True:**

**print("\nMenu:")**

**print("1. Register")**

**print("2. Login")**

**print("3. Exit")**

**choice = int(input("Enter your choice: "))**

**if choice == 1:**

**register\_user()**

**elif choice == 2:**

**login\_user()**

**elif choice == 3:**

**print("Exiting!")**

**break**

**else:**

**print("Invalid choice. Please choose a valid option.")**

**if \_\_name\_\_ == "\_\_main\_\_":**

**main()**

**Secure Login System using Shift Ciphers (Affine)**

**def shift\_cipher\_encrypt(message, a, b):**

**encrypted\_message = ""**

**for c in message:**

**if c.isalpha():**

**if c.isupper():**

**base = 'A'**

**else:**

**base = 'a'**

**shifted\_c = chr(((ord(c) - ord(base)) \* a + b) % 26 + ord(base))**

**encrypted\_message += shifted\_c**

**else:**

**encrypted\_message += c**

**return encrypted\_message**

**def shift\_cipher\_decrypt(message, a, b):**

**try:**

**inverse\_a = pow(a, -1, 26)**

**except ValueError:**

**return print("Key is not invertible!")**

**decrypted\_message = ""**

**for c in message:**

**if c.isalpha():**

**if c.isupper():**

**base = 'A'**

**else:**

**base = 'a'**

**shifted\_c = chr(((ord(c) - ord(base) - b) \* inverse\_a) % 26 + ord(base))**

**decrypted\_message += shifted\_c**

**else:**

**decrypted\_message += c**

**return decrypted\_message**

**stored\_users = {}**

**def register\_user():**

**username = input("Enter a username: ")**

**if username in stored\_users:**

**print("Username already exists!")**

**return**

**password = input("Enter a password: ")**

**a = int(input("Enter the additive key for encryption: "))**

**b = int(input("Enter the multiplicative key for encryption: "))**

**encrypted\_password = shift\_cipher\_encrypt(password, a, b)**

**stored\_users[username] = (encrypted\_password, a, b)**

**print("User registered successfully!")**

**def login\_user():**

**username = input("Enter your username: ")**

**if username not in stored\_users:**

**print("User does not exist!")**

**return**

**password = input("Enter your password: ")**

**encrypted\_password, a, b = stored\_users[username]**

**if encrypted\_password == shift\_cipher\_encrypt(password, a, b):**

**print("Login successful!")**

**else:**

**print("Incorrect password!")**

**def main():**

**while True:**

**print("\nMenu:")**

**print("1. Register")**

**print("2. Login")**

**print("3. Exit")**

**choice = int(input("Enter your choice: "))**

**if choice == 1:**

**register\_user()**

**elif choice == 2:**

**login\_user()**

**elif choice == 3:**

**print("Exiting!")**

**break**

**else:**

**print("Invalid choice. Please choose a valid option.")**

**if \_\_name\_\_ == "\_\_main\_\_":**

**main()**

**Shift Ciphers Implementation using Socket Programming**

**client.py**

**import socket**

**def additive\_cipher\_encrypt(message, key):**

**encrypted\_message = ""**

**for c in message:**

**if c.isalpha():**

**if c.isupper():**

**base = 'A'**

**else:**

**base = 'a'**

**shifted\_c = chr((ord(c) - ord(base) + key) % 26 + ord(base))**

**encrypted\_message += shifted\_c**

**else:**

**encrypted\_message += c**

**return encrypted\_message**

**def client():**

**client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)**

**client\_socket.connect(("localhost", 9999))**

**message = input("Enter the message to be sent to the server: ")**

**key = int(input("Enter key (integer) to encrypt the message: "))**

**encrypted\_message = additive\_cipher\_encrypt(message, key)**

**data = f"{encrypted\_message}|{key}"**

**client\_socket.send(data.encode('utf-8'))**

**result = client\_socket.recv(1024).decode('utf-8')**

**print("Decrypted message received from the server: ", result)**

**client\_socket.close()**

**if \_\_name\_\_ == "\_\_main\_\_":**

**client()**

**server.py**

**import socket**

**def additive\_cipher\_decrypt(message, key):**

**decrypted\_message = ""**

**for c in message:**

**if c.isalpha():**

**if c.isupper():**

**base = 'A'**

**else:**

**base = 'a'**

**shifted\_c = chr((ord(c) - ord(base) - key) % 26 + ord(base))**

**decrypted\_message += shifted\_c**

**else:**

**decrypted\_message += c**

**return decrypted\_message**

**def handle\_client(client\_socket):**

**try:**

**data = client\_socket.recv(1024).decode('utf-8')**

**encrypted\_message, key = data.split('|')**

**key = int(key)**

**decrypted\_message = additive\_cipher\_decrypt(encrypted\_message, key)**

**print("Decrypted message received from client: ", decrypted\_message)**

**client\_socket.send(decrypted\_message.encode('utf-8'))**

**finally:**

**client\_socket.close()**

**def start\_server():**

**server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)**

**server\_socket.bind(("localhost", 9999))**

**server\_socket.listen(5)**

**print("Server is listening on port 9999.")**

**while True:**

**client\_socket, address = server\_socket.accept()**

**print(f"Connection from {address} has been established.")**

**handle\_client(client\_socket)**

**if \_\_name\_\_ == "\_\_main\_\_":**

**start\_server()**