Cyber Security Lab Programs

Program 1

```
import hashlib
def hash_password(password):
    password_bytes=password.encode('utf-8')
    hash_object=hashlib.sha256(password_bytes)
    password_hash=hash_object.hexdigest()
    return password_hash
password=input("input your password:")
hashed_password=hash_password(password)
print(f"your hashed password is:{hashed_password}")
```

Program 2

```
import random
import string
def generate_password(length=8):
 characters = string.ascii_letters + string.digits + string.punctuation
 password = ''.join(random.choice(characters) for i in range(length))
 return password
password_length = int(input("Input the desired length of your password:")or 8)
password = generate_password(password_length)
print("Generated password is:", password)
```

Program 3

```
import re
def validate_password(password):
  if len(password) < 8:
      print("Password Should have atleast 8 characters")
      return False
  if not re.search(r'[A-Z]', password):
      print("Password should have atleast one uppercase letter")
      return False
  if not re.search(r'[a-z]', password):
      print("Password should contain atleast one lowercase letter")
      return False
  if not re.search(r'\d', password):
      print("password should have atleast one digit")
      return False
  if not re.search(r'[!@#$%^&*(),.?":{}|<>]', password):

      print("Password should have atleast one special character")
      return False
  return True
```

```
password = input("Input your password: ")
if validate_password(password):
    print("Valid Password.")
else:
    print("Password does not meet requirements.")
```

Program 4

```
import re
def check_password(password):
 length_regex = re.compile(r'^.{8,}$')
 uppercase_regex = re.compile(r'[A-Z]')
 lowercase_regex = re.compile(r'[a-z]')
 digit_regex = re.compile(r'\d')
 special_regex = re.compile(r'[\W_]')
 length_check = length_regex.search(password)
 uppercase_check = uppercase_regex.search(password)
 lowercase_check = lowercase_regex.search(password)
 digit_check = digit_regex.search(password)
 special_check = special_regex.search(password)
 if length_check and uppercase_check and lowercase_check and digit_check and
special_check:
   return True
 else:
   return False
with open(r'C:------') as f:
 for password in f:
   password = password.strip() # Remove newline character
   if check_password(password):
      print("Valid Password: "+password)
   else:
      print("Invalid: " + password)
```

Program 5

```
import re
def check_password_strength(password):
    score = 0
    suggestions = []
    if len(password) >= 8:
       score += 1
    else:
       suggestions.append("Password should be at least 8 characters long")
    if re.search(r"[A-Z]", password):
      score += 1
    else:
      suggestions.append("Password should contain at least one uppercase letter")
    if re.search(r"[a-z]", password):
      score += 1
    else:
        suggestions.append("Password should contain at least one lowercase letter")
    if re.search(r"\d", password):
        score += 1
    else:
        suggestions.append("Password should contain at least one numeric digit")
    if re.search(r"[!_@#$%^&*/(),.?\":{}|<>]", password):
        score += 1
    else:
        suggestions.append("Password should contain at least one special character
(!@#$%^&*(),.?\":{}|<>)")
    return score, suggestions
password = input("Input a password: ")
score, suggestions=check_password_strength(password)
print(score)
print(f"Password Score:{score}/5")
if suggestions:
    print("\n Suggestions to improve your password:")
    for suggestion in suggestions:
        print(f"-{suggestion}")
else:
    print("\n your password meets all criteria!")
```

Program 6

```
import requests
import hashlib
with open(r'') as f:
    for line in f:
        username, password = line.strip().split(',')
        password_hash = hashlib.sha1(password.encode('utf-8')).hexdigest().upper()
        response =
requests.get(f"https://api.pwnedpasswords.com/range/{password_hash[:5]}")
        if response.status_code == 200:
            hashes = [line.split(':') for line in response.text.splitlines()]
            for h, count in hashes:
                if password_hash[5:] == h:
                    print(f"Password for {username} has been leaked {count} times.")
                    break
        else:
            print(f"Could not check password for user {username}.")
```

Program 7

```
import itertools
import string
def bruteforce_attack(password):
    chars = string.printable.strip()
    attempts = 0
    for length in range(1, len(password) + 1):
        for guess in itertools.product(chars, repeat=length):
            attempts += 1
            guess = ''.join(guess)
            if guess == password:
                return (attempts, guess)
    return (attempts, None)
password = input("Input the password to crack: ")
attempts, guess = bruteforce_attack(password)
if guess:
    print(f"Password cracked in {attempts} attempts. The password is {guess}.")
else:
    print(f"Password not cracked after {attempts} attempts.")
```

Program 8

```
def encrypt(text, s):
    result = ""
    for i in range(len(text)):
        char = text[i]
        if char.isupper():
            result += chr((ord(char) + s - 65) % 26 + 65)
        # Encrypt lowercase characters
        else:
            result += chr((ord(char) + s - 97) % 26 + 97)
    return result
text = input("Enter the text: ")
s = int(input("Enter the shift between 1 to 25: "))
print("Text: " + text)
print("Shift: " + str(s))
print("Cipher: " + encrypt(text, s))
```

Program 9

```
message = 'GMTLIVrHIQS' # encrypted message
LETTERS = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
for key in range(len(LETTERS)):
    translated = ''
    for symbol in message:
        if symbol in LETTERS:
            num = LETTERS.find(symbol)
            num = num - key
            if num < 0:
                num = num + len(LETTERS)
            translated = translated + LETTERS[num]
        else:
            translated = translated + symbol
    print('Hacking key #%s: %s' % (key, translated))
```

Program 10

```
import rsa
def rsa_algo( password) :
   publickey, privatekey= rsa.newkeys(512)
   print("Public key: ",publickey)
   print("Private key:",privatekey)
   message = password.encode ('utf-8')
   ciphertext = rsa.encrypt (message, publickey)
   print("cipher text!", ciphertext)
   decrypted_message= rsa.decrypt (ciphertext , privatekey)
   print("decrypted message", decrypted_message.decode('utf -8'))
password = input ("Input your text")
rsa_algo(password)
```

Program 11

```
import base64
my_string = "Hello, World!"
encoded_string = base64.b64encode(my_string.encode("utf-8"))
print("encoded string:",encoded_string)
my_string = "SGVsbG8sIFdvcmxkIQ=="
decoded_string = base64.b64decode(my_string)
print("decoded string:",decoded_string.decode("utf-8"))
```

Program 12

```
from cryptography.fernet import Fernet
key=Fernet.generate_key()
fernet = Fernet(key)
msg=input("Enter the Message: ").encode()
encrypted_msg=fernet.encrypt(msg)
decrypted_msg=fernet.decrypt(encrypted_msg)
decrypted_msg=decrypted_msg.decode()
print("Original Message: ", msg.decode())
print("Encrypted Message: ", encrypted_msg)
print("Decrypted Message: ", decrypted_msg)
```