Lab 4 :

```
def encrypt_caesar(plaintext, shift):
    encrypted_text = ""
    for char in plaintext:
        if char.isalpha():  # Check if character is a letter
            shift_amount = shift % 26
            ascii_offset = 65 if char.isupper() else 97
            encrypted_char = chr((ord(char) - ascii_offset + shift_amount) % 26 + ascii_offset)
            encrypted_text += encrypted_char
        else:
            encrypted_text += char  # Non-alphabetic characters are added without change
    return encrypted_text

def decrypt_caesar(ciphertext, shift):
    return encrypt_caesar(ciphertext, -shift)

shift = 3
encrypted = Khoor, Zruog!

decrypted = decrypt_caesar(encrypted, shift)
print(f"Decrypted: {decrypted}")
```

Output :

```
Encrypted: Khoor, Zruog!
Decrypted: Hello, World!
```

Lab 5 :

```
def text_to_hex(text):
    hex_output = ""
    for char in text:
        hex_output += format(ord(char), "02x") + " "
    return hex_output.strip()


text_message = "Hello, World!"
hexadecimal_representation = text_to_hex(text_message)
print(f"Text: {text_message}")
print(f"Hexadecimal: {hexadecimal_representation}")
```

Output :

```
Text: Hello, World!
Hexadecimal: 48 65 6c 6c 6f 2c 20 57 6f 72 6c 64 21
```

Lab 7:

```
def encrypt_caesar(plaintext, shift):
    encrypted_text = ""
    for char in plaintext:
        if char.isalpha():  # Check if character is a letter
            shift_amount = shift % 26  # Ensure the shift is within the range of 0-25
            ascii_offset = 65 if char.isupper() else 97
            encrypted_char = chr((ord(char) - ascii_offset + shift_amount) % 26 + ascii_offset)
            encrypted_text += encrypted_char
        else:
            encrypted_text += char  # Non-alphabetic characters are added without change
    return encrypted_text

def decrypt_caesar(ciphertext, shift):
    return encrypt_caesar(ciphertext, -shift)

# Example usage:
plaintext = "Hello, World!"
shift = 3

encrypted = encrypt_caesar(plaintext, shift)
print(f"Encrypted: {encrypted}")

decrypted = decrypt_caesar(encrypted, shift)
print(f"Decrypted: {decrypted}")
```

Output:

```
Encrypted: Khoor, Zruog!
Decrypted: Hello, World!
```

Lab 8 :

```
def prime_checker(p):
    # Checks If the number entered is a Prime Number or not
    if p < 1:
        return -1
    elif p > 1:
        if p == 2:
            return 1
        for i in range(2, p):
            if p % i == 0:
                return -1
            return 1
def primitive_check(g, p, L):
    # Checks If The Entered Number Is A Primitive Root Or Not
    for i in range(1, p):
        L.append(pow(g, i) % p)
    for i in range(1, p):
        if L.count(i) > 1:
            L.clear()
            return -1
        return 1
l = []
while 1:
    P = int(input("Enter P : "))
    if prime_checker(P) == -1:
        print("Number Is Not Prime, Please Enter Again!")
        continue
    break

while 1:
    G = int(input(f"Enter The Primitive Root Of {P} : "))
```

```
        if primitive_check(G, P, l) == -1:
                print(f"Number Is Not A Primitive Root Of {P}, Please Try Again!")
                continue
        break
x1, x2 = int(input("Enter The Private Key Of User 1 : ")), int(
        input("Enter The Private Key Of User 2 : "))
while 1:
        if x1 >= P or x2 >= P:
                print(f"Private Key Of Both The Users Should Be Less Than {P}!")
                continue
        break
y1, y2 = pow(G, x1) % P, pow(G, x2) % P

# Generate Secret Keys
k1, k2 = pow(y2, x1) % P, pow(y1, x2) % P

print(f"\nSecret Key For User 1 Is {k1}\nSecret Key For User 2 Is {k2}\n")

if k1 == k2:
        print("Keys Have Been Exchanged Successfully")
else:
        print("Keys Have Not Been Exchanged Successfully")
```

Output :

```
The value of P : 23
The value of G : 9
The private key a for Alice : 4
The private key b for Bob : 3
Secret key for the Alice is : 9
Secret Key for the Bob is : 9
```

Lab 2:

```
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)
            print(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"if password not cracked after {attempts} attempts.")
```

Output :

```
k`
k{
k|
k}
k~
```

l0

l1

l2

l3

l4

l5

l6

l7

l8

l9

la

Password cracked in 2079 attempts. The password is la

Lab 3:

```
import hashlib

# List of commonly used passwords and their variations
common_passwords = [
    "password", "password123", "letmein", "qwerty",
    "123456", "abc123", "admin", "welcome", "monkey", "sunshine"
]

password_variations = [
    "", "123", "1234", "12345", "123456", "1", "@", "S"
]

# Hash of the password to be attacked
hashed_password = hashlib.sha256(b"password123").hexdigest()

# Try out all possible combinations of common passwords and their variations
password_found = False

for password in common_passwords:
    for variation in password_variations:
        possible_password = password + variation
        hashed_possible_password = hashlib.sha256(possible_password.encode()).hexdigest()

        if hashed_possible_password == hashed_password:
            print(f"Password found: {possible_password}")
            password_found = True
            break

    if password_found:
        break
```

```
if not password_found:
    print("Password not found")
```

Output :

Password found: password123

Lab 6:

```
from tkinter import *
from tkinter import messagebox
import base64


def encrypt():
    password = code.get()
    if password == "1234":
        screen1 = Toplevel(screen)
        screen1.title("Encryption")
        screen1.geometry("400x200")
        screen1.configure(bg="red")
        message = text1.get(1.0, END)
        encode_message = message.encode("ascii")
        base64_bytes = base64.b64encode(encode_message)
        encrypt = base64_bytes.decode("ascii")
        Label(screen1, text="ENCRYPT", font="arial", fg="white", bg="red").place(x=10, y=0)
        text2 = Text(screen1, font="robote 10", bg="white", relief=GROOVE, wrap=WORD, bd=0)
        text2.place(x=10, y=40, width=380, height=150)
        text2.insert(END, encrypt)
    elif password == "":
        messagebox.showerror("Encryption", "Input password")
    elif password != "1234":
        messagebox.showerror("Encryption", "Invalid password")


def decrypt():
    password = code.get()
    if password == "1234":
        screen2 = Toplevel(screen)
        screen2.title("Decryption")
        screen2.geometry("400x200")
```

```
        screen2.configure(bg="green")
        message = text1.get(1.0, END)
        decode_message = message.encode("ascii")
        base64_bytes = base64.b64decode(decode_message)
        decrypt = base64_bytes.decode("ascii")
        Label(screen2, text="DECRYPT", font="arial", fg="white", bg="green").place(x=10, y=0)
        text2 = Text(screen2, font="robote 10", bg="white", relief=GROOVE, wrap=WORD, bd=0)
        text2.place(x=10, y=40, width=380, height=150)
        text2.insert(END, decrypt)
    elif password == "":
        messagebox.showerror("Decryption", "Input password")
    elif password != "1234":
        messagebox.showerror("Decryption", "Invalid password")


def main_screen():
    global text1, code, screen
    screen = Tk()
    screen.geometry("500x500")
    # Icon
    image_icon = PhotoImage(file="favicon.ico.png")
    screen.iconphoto(False, image_icon)
    screen.title("Secrets")


    def reset():
        code.set("")
        text1.delete(1.0, END)


    Label(text="Enter text for encryption and decryption", fg="black", font=("calibri", 13)).place(x=10,
y=10)
    text1 = Text(font="robote 20", bg="white", relief=GROOVE, wrap=WORD, bd=0)
    text1.place(x=10, y=50, width=355, height=100)
```

```
    Label(text="Enter secret key for encryption and decryption", fg="black", font=("calibri",
13)).place(x=10, y=170)

    code = StringVar()

    Entry(textvar=code, width=19, bd=0, font=("arial", 25)).place(x=10, y=200)


    Button(text="ENCRYPT", height=2, width=23, bg="red", fg="white", bd=0,
command=encrypt).place(x=10, y=250)

    Button(text="DECRYPT", height=2, width=23, bg="green", fg="white", bd=0,
command=decrypt).place(x=200, y=250)

    Button(text="RESET", height=2, width=50, bg="blue", fg="white", bd=0,
command=reset).place(x=10, y=300)


    screen.mainloop()


main_screen()
```