```
import math

def is_coprime(a, b):
    return math.gcd(a, b) == 1

def mod_inverse(a, m):
    for x in range(1, m):
        if (a * x) % m == 1:
            return x
    return None

def generate_keypair(p, q):
    n = p * q
    phi_n = (p - 1) * (q - 1)

    e = 2
    while e < phi_n:
        if is_coprime(e, phi_n):
            break
        e += 1

    d = mod_inverse(e, phi_n)

    return ((n, e), (n, d))

def encrypt(public_key, plaintext):
    n, e = public_key
    return (plaintext ** e) % n

def decrypt(private_key, ciphertext):
    n, d = private_key
    return (ciphertext ** d) % n

def main():
    p = int(input("Enter the first prime number (p): "))
    q = int(input("Enter the second prime number (q): "))

    public_key, private_key = generate_keypair(p, q)

    print("Public key (n, e):", public_key)
    print("Private key (n, d):", private_key)

    choice = input("Enter 'E' to encrypt or 'D' to decrypt: ").upper()

    if choice == 'E':
        plaintext = int(input("Enter the plaintext to encrypt: "))
        ciphertext = encrypt(public_key, plaintext)
        print("The encrypted ciphertext is:", ciphertext)
    elif choice == 'D':
        ciphertext = int(input("Enter the ciphertext to decrypt: "))
        decrypted_text = decrypt(private_key, ciphertext)
        print("The decrypted plaintext is:", decrypted_text)
    else:
        print("Invalid choice. Please enter 'E' or 'D.")

if __name__ == "__main__":
  main()
```

```
Enter the first prime number (p): 61
Enter the second prime number (q): 53
Public key (n, e): (3233, 7)
Private key (n, d): (3233, 1783)
Enter 'E' to encrypt or 'D' to decrypt: d
Enter the ciphertext to decrypt: 1754
The decrypted plaintext is: 45
```