```python
#CS-ASSIGNMENT 1 - RSA ALGORITHM
#HARSH MEHTA
#TA41
#BATCH B
import random

def gcd(a, b):
    while b != 0:
        a, b = b, a % b
    return a

def extended_gcd(a, b):
    if a == 0:
        return (b, 0, 1)
    else:
        gcd, x, y = extended_gcd(b % a, a)
        return (gcd, y - (b // a) * x, x)

def mod_inverse(a, m):
    gcd, x, y = extended_gcd(a, m)
    if gcd != 1:
        raise Exception('Modular inverse does not exist')
    else:
        return x % m

def generate_keypair(p, q):
    n = p * q
    phi = (p - 1) * (q - 1)
    e = random.randrange(1, phi)
    g = gcd(e, phi)
    while g != 1:
        e = random.randrange(1, phi)
        g = gcd(e, phi)
    d = mod_inverse(e, phi)
    return ((e, n), (d, n))

def encrypt(public_key, plaintext):
    e, n = public_key
    cipher = [pow(ord(char), e, n) for char in plaintext]
    return cipher

def decrypt(private_key, ciphertext):
    d, n = private_key
    plain = [chr(pow(char, d, n)) for char in ciphertext]
    return ''.join(plain)

if __name__ == "__main__":
    p = 61
    q = 53
    public_key, private_key = generate_keypair(p, q)
    print("Public key:", public_key)
    print("Private key:", private_key)

    message = input("Enter the message : ")
    encrypted_message = encrypt(public_key, message)
    print("Encrypted message:", encrypted_message)

    decrypted_message = decrypt(private_key, encrypted_message)
    print("Decrypted message:", decrypted_message)
```

```
Public key: (3037, 3233)
Private key: (2293, 3233)
Enter the message : hi harsh mehta
Encrypted message: [2072, 1006, 2228, 2072, 598, 1253, 409, 2072, 2228, 1207, 2358, 2072, 1349, 598]
Decrypted message: hi harsh mehta
```