main.c

Run

Output

Clear

```c
#include <stdio.h>

// Extended Euclidean Algorithm
int extended_gcd(int a, int b, int *x, int *y) {
    if (b == 0) {
        *x = 1;
        *y = 0;
        return a;
    }
    int x1, y1;
    int gcd = extended_gcd(b, a % b, &x1, &y1);
    *x = y1;
    *y = x1 - (a / b) * y1;
    return gcd;
}

// Modular inverse: returns d such that (e * d)
    % phi = 1
```

```
Public Key: (e = 31, n = 3599)
Private Key: d = 3031
Plaintext: 123
Encrypted : 733
Decrypted : 123


=== Code Execution Successful ===
```

main.c

Run

Output

Clear

```
    }

    printf("Public Key: (e = %d, n = %d)\n", e,
        n);
    printf("Private Key: d = %d\n", d);

    // Test encryption/decryption
    int plaintext = 123;
    int ciphertext = encrypt(plaintext, e, n);
    int decrypted = decrypt(ciphertext, d, n);

    printf("Plaintext: %d\n", plaintext);
    printf("Encrypted : %d\n", ciphertext);
    printf("Decrypted : %d\n", decrypted);

    return 0;
}
```

```
Public Key: (e = 31, n = 3599)
Private Key: d = 3031
Plaintext: 123
Encrypted : 733
Decrypted : 123


=== Code Execution Successful ===
```