



# VIT<sup>®</sup>

## Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

**B.Tech. Winter Semester 2024-25**  
**School Of Computer Science and Engineering**  
**(SCOPE)**

# Digital Assignment - II

## Cryptography and Network Security Lab

Apurva Mishra: 22BCE2791

Date: 16 February, 2025

### Contents

<b>1 Fermat's Theorem</b>	<b>2</b>
1.1 Code	2
1.2 Output	3
<b>2 Euler' Theorem</b>	<b>3</b>
2.1 Code	3
2.2 Output	5
<b>3 Euclidian Algorithm</b>	<b>5</b>
3.1 Code	5
3.2 Output	6

# 1 Fermat's Theorem

## 1.1 Code

### Code 0: main.c

```
1 // fermats_theorem.c
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <stdbool.h>
5
6 // Function to check if a number is prime.
7 bool isPrime(int n) {
8     if (n <= 1) return false;
9     if (n <= 3) return true;
10    if (n % 2 == 0 || n % 3 == 0) return false;
11    for (int i = 5; i * i <= n; i += 6) {
12        if (n % i == 0 || n % (i + 2) == 0)
13            return false;
14    }
15    return true;
16 }
17
18 // Fast modular exponentiation: computes (base^exp) mod mod.
19 long long modExp(long long base, long long exp, int mod) {
20     long long result = 1;
21     base = base % mod;
22     while(exp > 0) {
23         if(exp % 2 == 1)
24             result = (result * base) % mod;
25         exp = exp >> 1; // divide exp by 2
26         base = (base * base) % mod;
27     }
28     return result;
29 }
30
31 int main(void) {
32     int a, p;
33     printf("Fermat's Little Theorem Checker\n");
34     printf("Enter an integer a: ");
35     if (scanf("%d", &a) != 1) {
36         fprintf(stderr, "Invalid input.\n");
37         return 1;
38     }
39     printf("Enter a prime number p: ");
40     if (scanf("%d", &p) != 1) {
41         fprintf(stderr, "Invalid input.\n");
42         return 1;
43     }
44
45     if (!isPrime(p)) {
46         printf("Error: %d is not a prime number.\n", p);
47         return 1;
48     }
```

```

48     }
49
50     if (a % p == 0) {
51         printf("Note: a is divisible by p. Fermat's theorem applies only if
a is not divisible by p.\n");
52         return 1;
53     }
54
55     // Fermat's Little Theorem: a^(p-1) mod p should equal 1.
56     long long result = modExp(a, p - 1, p);
57     printf("Computed: %d^(%d-1) mod %d = %lld\n", a, p, p, result);
58
59     if (result == 1)
60         printf("Fermat's Little Theorem holds for a = %d and p = %d.\n",
a, p);
61     else
62         printf("Fermat's Little Theorem does not hold (unexpected result).\n"
63
64         return 0;
65 }
66

```

## 1.2 Output

```

da/ass2/q1 via C v16.0.0-clang
> just run
zig cc main.c -o main
./main
Fermat's Little Theorem Checker
Enter an integer a: 6
Enter a prime number p: 7
Computed: 6^(7-1) mod 7 = 1
Fermat's Little Theorem holds for a = 6 and p = 7.

da/ass2/q1 via C v16.0.0-clang took 5s
> 

```

## 2 Euler' Theorem

### 2.1 Code

#### Code 0: main.c

```

1 // euler_theorem.c
2 #include <stdio.h>
3 #include <stdlib.h>
4
5 // Function to compute the Greatest Common Divisor using the Euclidean

```

algorithm.

```
6  int gcd(int a, int b) {
7      while(b != 0) {
8          int temp = b;
9          b = a % b;
10         a = temp;
11     }
12     return a;
13 }
14
15 // Fast modular exponentiation: computes (base^exp) mod mod.
16 long long modExp(long long base, long long exp, int mod) {
17     long long result = 1;
18     base = base % mod;
19     while(exp > 0) {
20         if(exp % 2 == 1)
21             result = (result * base) % mod;
22         exp = exp >> 1;
23         base = (base * base) % mod;
24     }
25     return result;
26 }
27
28 // Function to compute Euler's Totient Function,  $\phi(n)$ 
29 int phi(int n) {
30     int result = n;
31     for (int i = 2; i * i <= n; i++) {
32         if (n % i == 0) {
33             while(n % i == 0)
34                 n /= i;
35             result -= result / i;
36         }
37     }
38     if(n > 1)
39         result -= result / n;
40     return result;
41 }
42
43 int main(void) {
44     int a, n;
45     printf("Euler's Theorem Checker\n");
46     printf("Enter an integer a: ");
47     if(scanf("%d", &a) != 1) {
48         fprintf(stderr, "Invalid input.\n");
49         return 1;
50     }
51     printf("Enter a positive integer n: ");
52     if(scanf("%d", &n) != 1 || n <= 0) {
53         fprintf(stderr, "Invalid input.\n");
54         return 1;
55     }
56
57     int g = gcd(a, n);
```

```

58     printf("gcd(%d, %d) = %d\n", a, n, g);
59
60     if(g != 1) {
61         printf("Case ii: Since gcd(a, n)  $\neq$  1, Euler's Theorem does not
apply.\n");
62     } else {
63         // Case i: When a and n are relatively prime.
64         int totient = phi(n);
65         printf("Euler's Totient Function  $\phi$ (%d) = %d\n", n, totient);
66         long long result = modExp(a, totient, n);
67         printf("Computed: %d^( $\phi$ (%d)) mod %d = %lld\n", a, n, n, result);
68         if(result == 1)
69             printf("Euler's Theorem holds for a = %d and n = %d.\n", a, n);
70         else
71             printf("Euler's Theorem does not hold (unexpected result).\n");
72     }
73
74     return 0;
75 }

```

## 2.2 Output

```

da/ass2/q2 via C v16.0.0-clang
) just run
zig cc main.c -o main
./main
Euler's Theorem Checker
Enter an integer a: 6
Enter a positive integer n: 7
gcd(6, 7) = 1
Euler's Totient Function  $\phi$ (7) = 6
Computed: 6^( $\phi$ (7)) mod 7 = 1
Euler's Theorem holds for a = 6 and n = 7.

da/ass2/q2 via C v16.0.0-clang took 9s
) |

```

## 3 Euclidian Algorithm

### 3.1 Code

Code 0: main.c

```

1 // euclidean_algorithm.c
2 #include <stdio.h>
3 #include <stdlib.h>
4

```

```

5 // Euclidean Algorithm to compute gcd of two numbers.
6 int gcd(int a, int b) {
7     while(b != 0) {
8         int temp = b;
9         b = a % b;
10        a = temp;
11    }
12    return a;
13 }
14
15 int main(void) {
16     int num1, num2;
17     printf("Euclidean Algorithm for GCD\n");
18     printf("Enter first integer: ");
19     if(scanf("%d", &num1) != 1) {
20         fprintf(stderr, "Invalid input.\n");
21         return 1;
22     }
23     printf("Enter second integer: ");
24     if(scanf("%d", &num2) != 1) {
25         fprintf(stderr, "Invalid input.\n");
26         return 1;
27     }
28
29     int result = gcd(num1, num2);
30     printf("gcd(%d, %d) = %d\n", num1, num2, result);
31
32     return 0;
33 }
34

```

## 3.2 Output

```

da/ass2/q3 via C v16.0.0-clang
> just run
zig cc main.c -o main
./main
Euclidean Algorithm for GCD
Enter first integer: 64
Enter second integer: 8
gcd(64, 8) = 8

da/ass2/q3 via C v16.0.0-clang took 7s
> |

```