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

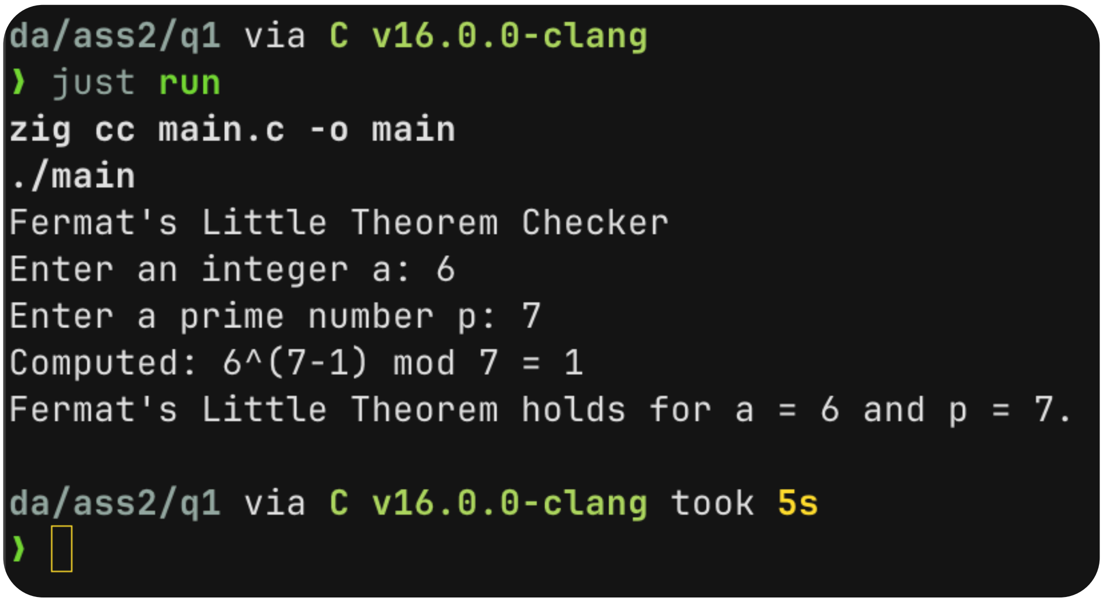
1 Fermat’s Theorem

1.1 Code

Code 0: main.c

```c
// fermats_theorem.c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

// Function to check if a number is prime.
bool isPrime(int n) {
    if (n <= 1) return false;
    if (n <= 3) return true;
    if (n % 2 == 0 || n % 3 == 0) return false;
    for (int i = 5; i * i <= n; i += 6) {
        if (n % i == 0 || n % (i + 2) == 0)
            return false;
    }
    return true;
}

// Fast modular exponentiation: computes (base^exp) mod mod.
long long modExp(long long base, long long exp, int mod) {
    long long result = 1;
    base = base % mod;
    while(exp > 0) {
        if(exp % 2 == 1)
            result = (result * base) % mod;
        exp = exp >> 1;  // divide exp by 2
        base = (base * base) % mod;
    }
    return result;
}

int main(void) {
    int a, p;
    printf("Fermat's Little Theorem Checker\n");
    printf("Enter an integer a: ");
    if (scanf("%d", &a) != 1) {
        fprintf(stderr, "Invalid input.\n");
        return 1;
    }
    printf("Enter a prime number p: ");
    if (scanf("%d", &p) != 1) {
        fprintf(stderr, "Invalid input.\n");
        return 1;
    }

    if (!isPrime(p)) {
        printf("Error: %d is not a prime number.\n", p);
        return 1;
    }

    if (a % p == 0) {
        printf("Note: a is divisible by p. Fermat's theorem applies only if a is not divisible by p.\n");
        return 1;
    }

    // Fermat's Little Theorem: a^(p-1) mod p should equal 1.
    long long result = modExp(a, p - 1, p);
    printf("Computed: %d^(%d-1) mod %d = %lld\n", a, p, p, result);

    if (result == 1)
         printf("Fermat's Little Theorem holds for a = %d and p = %d.\n", a, p);
    else
        printf("Fermat's Little Theorem does not hold (unexpected result).\n"

    return 0;
}

```

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

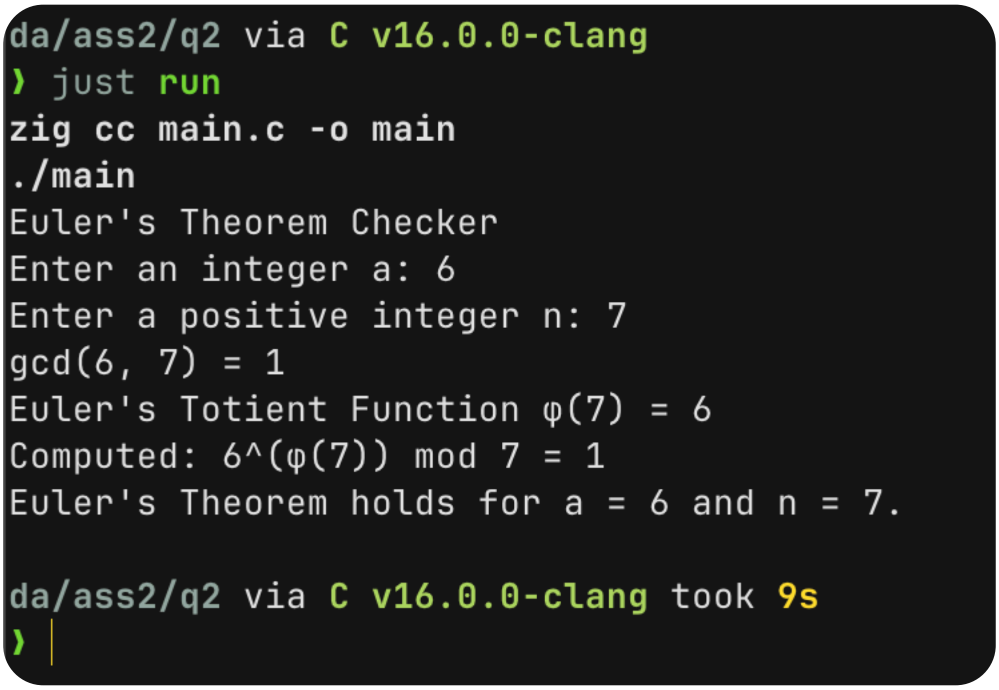
2.1 Code

Code 0: main.c

```c
// euler_theorem.c
#include <stdio.h>
#include <stdlib.h>

// Function to compute the Greatest Common Divisor using the Euclidean
```

```c
algorithm.
int gcd(int a, int b) {
    while(b != 0) {
        int temp = b;
        b = a % b;
        a = temp;
    }
    return a;
}

// Fast modular exponentiation: computes (base^exp) mod mod.
long long modExp(long long base, long long exp, int mod) {
    long long result = 1;
    base = base % mod;
    while(exp > 0) {
        if(exp % 2 == 1)
            result = (result * base) % mod;
        exp = exp >> 1;
        base = (base * base) % mod;
    }
    return result;
}

// Function to compute Euler's Totient Function, φ(n)
int phi(int n) {
    int result = n;
    for (int i = 2; i * i <= n; i++) {
        if (n % i == 0) {
            while(n % i == 0)
                n /= i;
            result -= result / i;
        }
    }
    if(n > 1)
        result -= result / n;
    return result;
}

int main(void) {
    int a, n;
    printf("Euler's Theorem Checker\n");
    printf("Enter an integer a: ");
    if(scanf("%d", &a) != 1) {
        fprintf(stderr, "Invalid input.\n");
        return 1;
    }
    printf("Enter a positive integer n: ");
    if(scanf("%d", &n) != 1 || n <= 0) {
        fprintf(stderr, "Invalid input.\n");
        return 1;
    }

    int g = gcd(a, n);
    printf("gcd(%d, %d) = %d\n", a, n, g);

    if(g != 1) {
          printf("Case ii: Since gcd(a, n) ≠ 1, Euler's Theorem does not apply.\n");
    } else {
        // Case i: When a and n are relatively prime.
        int totient = phi(n);
        printf("Euler's Totient Function φ(%d) = %d\n", n, totient);
        long long result = modExp(a, totient, n);
        printf("Computed: %d^(φ(%d)) mod %d = %lld\n", a, n, n, result);
        if(result == 1)
            printf("Euler's Theorem holds for a = %d and n = %d.\n", a, n);
        else
            printf("Euler's Theorem does not hold (unexpected result).\n");
    }

    return 0;
}
```

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 φ(7) = 6
Computed: 6^(φ(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

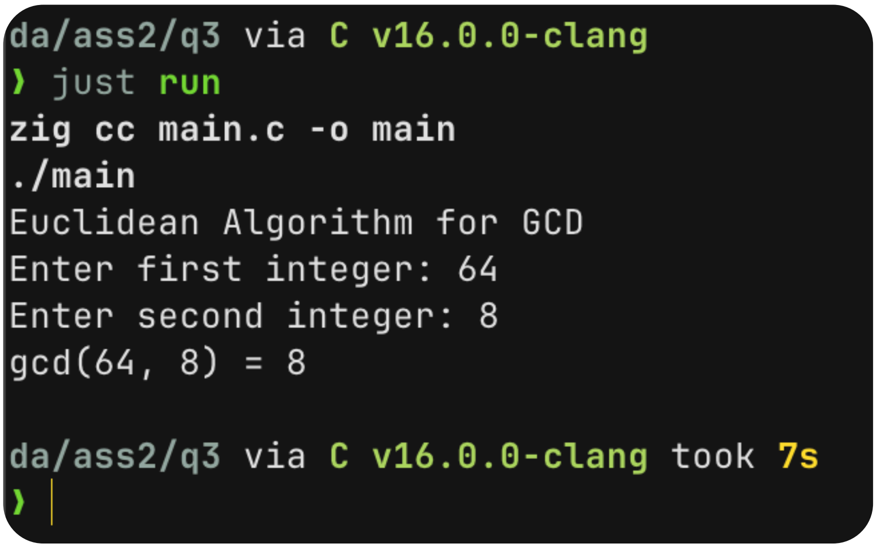
3.1 Code

Code 0: main.c

```c
// euclidean_algorithm.c
#include <stdio.h>
#include <stdlib.h>

```

```c
// Euclidean Algorithm to compute gcd of two numbers.
int gcd(int a, int b) {
    while(b != 0) {
        int temp = b;
        b = a % b;
        a = temp;
    }
    return a;
}

int main(void) {
    int num1, num2;
    printf("Euclidean Algorithm for GCD\n");
    printf("Enter first integer: ");
    if(scanf("%d", &num1) != 1) {
        fprintf(stderr, "Invalid input.\n");
        return 1;
    }
    printf("Enter second integer: ");
    if(scanf("%d", &num2) != 1) {
        fprintf(stderr, "Invalid input.\n");
        return 1;
    }

    int result = gcd(num1, num2);
    printf("gcd(%d, %d) = %d\n", num1, num2, result);

    return 0;
}

```

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
❯
```