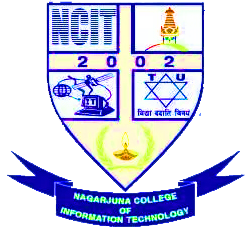
**LAB REPORT**

**OF**

**CRYPTOGRAPHY ( CSC 316)**

****

**Submitted to**

**NAGARJUNA COLLEGE OF IT**

***(AFFILATED TO TRIBHUVAN UNIVERSITY)***

**Shankamul, Lalitpur- 09**

**Submitted By**

Hari Prasad Gyawali

***College Roll Number* : 06**

**Program : B.Sc. CSIT**

**Semester : IV**

**Lab 01**

1. **Write a program to implement Shift Cipher.**

#include <stdio.h>

#include <string.h>

void shift\_cipher(char text[], int shift) {

int length = strlen(text);

for (int i = 0; i < length; i++) {

if (isalpha(text[i])) {

// Encrypt uppercase letters

if (isupper(text[i])) {

text[i] = (text[i] + shift - 'A') % 26 + 'A';

}

// Encrypt lowercase letters

else {

text[i] = (text[i] + shift - 'a') % 26 + 'a';

}

}

}

}

int main() {

char plaintext[100];

int shift\_value;

// Input

printf("Enter the plaintext: ");

fgets(plaintext, sizeof(plaintext), stdin);

printf("Enter the shift value: ");

scanf("%d", &shift\_value);

// Remove newline character from fgets

plaintext[strcspn(plaintext, "\n")] = '\0';

// Encryption

shift\_cipher(plaintext, shift\_value);

// Output

printf("Encrypted: %s\n", plaintext);

// Your name and roll number

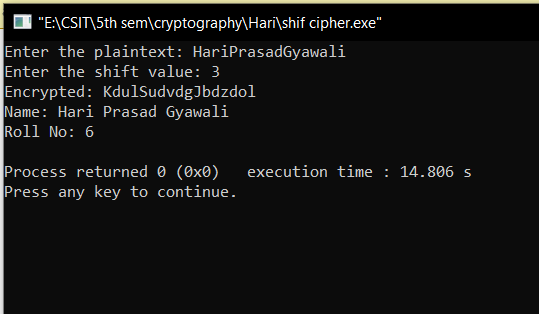
printf("Name: Hari Prasad Gyawali\n");

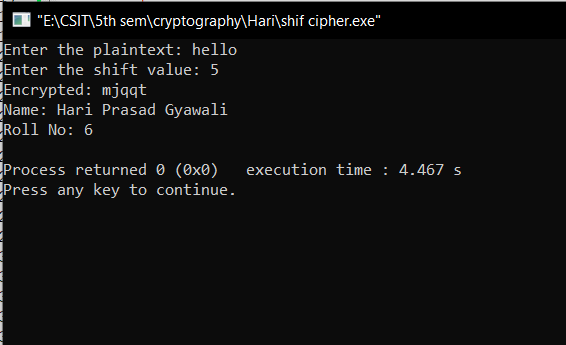
printf("Roll No: 6\n");

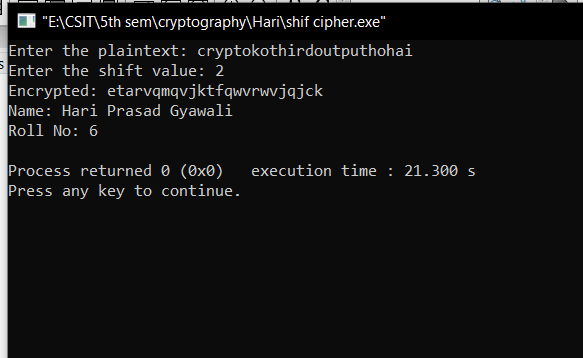
return 0;

}

Output:







**Lab 02**

1. **Write a program to implement Caeser Cipher.**

#include <stdio.h>

// Function to perform Caesar Cipher encryption

void caesar\_cipher(char text[], int shift) {

for (int i = 0; text[i] != '\0'; i++) {

// Encrypt uppercase letters

if (text[i] >= 'A' && text[i] <= 'Z') {

text[i] = (text[i] + shift - 'A') % 26 + 'A';

}

// Encrypt lowercase letters

else if (text[i] >= 'a' && text[i] <= 'z') {

text[i] = (text[i] + shift - 'a') % 26 + 'a';

}

}

}

int main() {

char plaintext[100];

int shift\_value;

char key[100];

// First encryption

printf("Enter the first plaintext: ");

gets(plaintext);

printf("Enter the shift value for the first encryption: ");

scanf("%d", &shift\_value);

getchar();

printf("Enter the key for the first encryption: ");

gets(key);

caesar\_cipher(plaintext, shift\_value);

printf("Encrypted (1): %s\n", plaintext);

// Second encryption

printf("\nEnter the second plaintext: ");

gets(plaintext);

printf("Enter the shift value for the second encryption: ");

scanf("%d", &shift\_value);

getchar();

printf("Enter the key for the second encryption: ");

gets(key);

caesar\_cipher(plaintext, shift\_value);

printf("Encrypted (2): %s\n", plaintext);

// Third encryption

printf("\nEnter the third plaintext: ");

gets(plaintext);

printf("Enter the shift value for the third encryption: ");

scanf("%d", &shift\_value);

getchar();

printf("Enter the key for the third encryption: ");

gets(key);

caesar\_cipher(plaintext, shift\_value);

printf("Encrypted (3): %s\n", plaintext);

// Your name and roll number

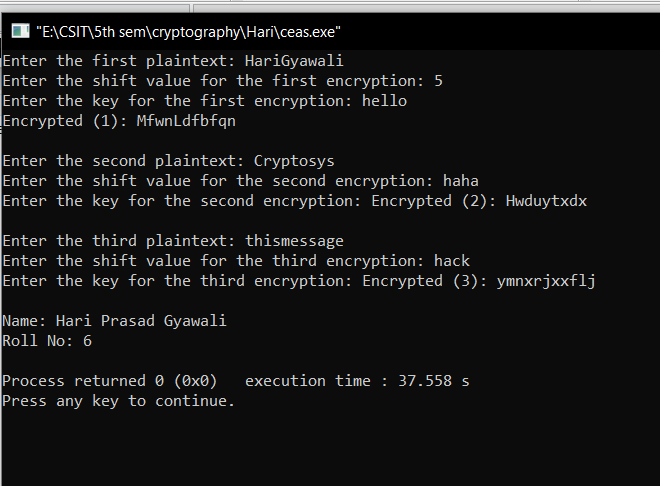
printf("\nName: Hari Prasad Gyawali\n");

printf("Roll No: 6\n");

return 0;

}

Output:



**Lab 03**

1. **Write a program to implement Vigenere Cipher.**

#include <stdio.h>

#include <string.h>

void vigenere\_cipher(char text[], char key[]) {

int text\_length = strlen(text);

int key\_length = strlen(key);

for (int i = 0, j = 0; i < text\_length; i++) {

if (isalpha(text[i])) {

int shift = key[j % key\_length] - 'A';

if (isupper(text[i])) {

text[i] = (text[i] + shift - 'A') % 26 + 'A';

} else {

text[i] = (text[i] + shift - 'a') % 26 + 'a';

}

j++;

}

}

}

int main() {

char plaintext[100];

char key[100];

// Input

printf("Enter the plaintext: ");

gets(plaintext);

printf("Enter the key: ");

gets(key);

// Encryption

vigenere\_cipher(plaintext, key);

// Output

printf("Encrypted: %s\n", plaintext);

// Additional inputs for three different words

char input2[100], input3[100];

printf("Enter another plaintext: ");

gets(input2);

printf("Enter another key: ");

gets(key);

vigenere\_cipher(input2, key);

printf("Encrypted: %s\n", input2);

printf("Enter one more plaintext: ");

gets(input3);

printf("Enter one more key: ");

gets(key);

vigenere\_cipher(input3, key);

printf("Encrypted: %s\n", input3);

// Your name and roll number

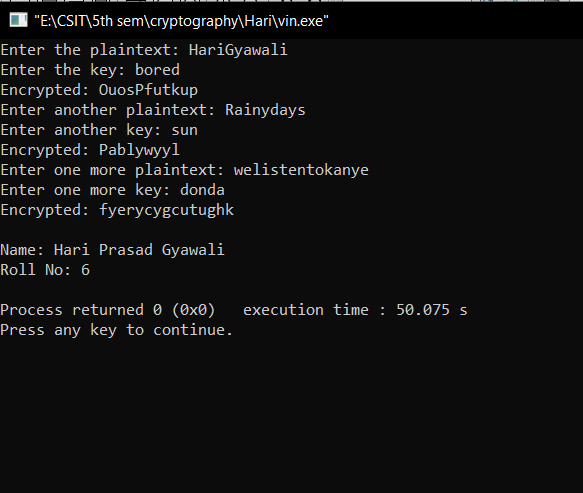
printf("\nName: Hari Prasad Gyawali\n");

printf("Roll No: 6\n");

return 0;

}

Output:



**Lab 04**

1. **Write a program to implement PlayFair Cipher.**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define SIZE 5

void generateKeyMatrix(char key[], char keyMatrix[SIZE][SIZE]) {

int k, flag = 0, \*dicty;

// Create a dictionary to check for repeated characters

dicty = (int \*)calloc(26, sizeof(int));

for (k = 0; k < strlen(key); k++) {

if (key[k] != 'J') {

if (dicty[key[k] - 65] == 0) {

keyMatrix[flag / 5][flag % 5] = key[k];

dicty[key[k] - 65] = 1;

flag++;

}

}

}

// Fill the remaining cells with the alphabet

for (k = 0; k < 26; k++) {

if (dicty[k] == 0) {

keyMatrix[flag / 5][flag % 5] = (char)(k + 65);

flag++;

}

}

free(dicty);

}

void playfair\_cipher(char message[], char keyMatrix[SIZE][SIZE]) {

int i, j, k = 0, len = strlen(message);

for (k = 0; k < len; k += 2) {

// Find the coordinates of each digraph

int row1, col1, row2, col2;

for (i = 0; i < SIZE; i++) {

for (j = 0; j < SIZE; j++) {

if (message[k] == keyMatrix[i][j]) {

row1 = i;

col1 = j;

}

if (message[k + 1] == keyMatrix[i][j]) {

row2 = i;

col2 = j;

}

}

}

// Apply the rules of the Playfair Cipher

if (row1 == row2) {

message[k] = keyMatrix[row1][(col1 + 1) % SIZE];

message[k + 1] = keyMatrix[row2][(col2 + 1) % SIZE];

} else if (col1 == col2) {

message[k] = keyMatrix[(row1 + 1) % SIZE][col1];

message[k + 1] = keyMatrix[(row2 + 1) % SIZE][col2];

} else {

message[k] = keyMatrix[row1][col2];

message[k + 1] = keyMatrix[row2][col1];

}

}

}

int main() {

char key[100], message[100];

char keyMatrix[SIZE][SIZE];

// Input

printf("Enter the key: ");

fgets(key, sizeof(key), stdin);

key[strcspn(key, "\n")] = '\0'; // remove the newline character

printf("Enter the message: ");

fgets(message, sizeof(message), stdin);

message[strcspn(message, "\n")] = '\0'; // remove the newline character

// Generate key matrix

generateKeyMatrix(key, keyMatrix);

// Encryption

playfair\_cipher(message, keyMatrix);

// Output

printf("Encrypted: %s\n", message);

// Additional inputs for three different words

char input2[100], input3[100];

printf("Enter another message: ");

fgets(input2, sizeof(input2), stdin);

input2[strcspn(input2, "\n")] = '\0';

playfair\_cipher(input2, keyMatrix);

printf("Encrypted: %s\n", input2);

printf("Enter one more message: ");

fgets(input3, sizeof(input3), stdin);

input3[strcspn(input3, "\n")] = '\0';

playfair\_cipher(input3, keyMatrix);

printf("Encrypted: %s\n", input3);

// Your name and roll number

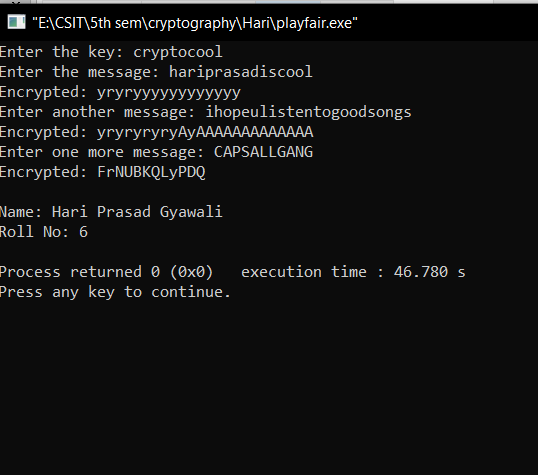
printf("\nName: Hari Prasad Gyawali\n");

printf("Roll No: 6\n");

return 0;

}

Output:



**Lab 05**

1. **Write a program to implement railfence Cipher.**

#include <stdio.h>

#include <string.h>

void railfence\_cipher(char text[], int key) {

int len = strlen(text);

char encrypted[len];

int r = key;

int c = len / key;

if (len % key != 0) {

c++;

}

char matrix[r][c];

int k = 0;

// Fill the matrix with characters from the text

for (int i = 0; i < r; i++) {

for (int j = 0; j < c; j++) {

if (k < len) {

matrix[i][j] = text[k++];

} else {

matrix[i][j] = 'X'; // Padding with 'X' if needed

}

}

}

// Read the matrix in a zigzag pattern to get the encrypted text

k = 0;

for (int j = 0; j < c; j++) {

for (int i = 0; i < r; i++) {

if (matrix[i][j] != 'X') {

encrypted[k++] = matrix[i][j];

}

}

}

encrypted[k] = '\0'; // Null-terminate the encrypted text

// Print the encrypted text

printf("Encrypted: %s\n", encrypted);

}

int main() {

char plaintext[100];

int key;

// Input and encryption 1

printf("Enter the first plaintext: ");

fgets(plaintext, sizeof(plaintext), stdin);

plaintext[strcspn(plaintext, "\n")] = '\0'; // remove the newline character

printf("Enter the key for the first encryption: ");

scanf("%d", &key);

getchar(); // Consume the newline character

railfence\_cipher(plaintext, key);

// Input and encryption 2

printf("\nEnter the second plaintext: ");

fgets(plaintext, sizeof(plaintext), stdin);

plaintext[strcspn(plaintext, "\n")] = '\0'; // remove the newline character

printf("Enter the key for the second encryption: ");

scanf("%d", &key);

getchar(); // Consume the newline character

railfence\_cipher(plaintext, key);

// Input and encryption 3

printf("\nEnter the third plaintext: ");

fgets(plaintext, sizeof(plaintext), stdin);

plaintext[strcspn(plaintext, "\n")] = '\0'; // remove the newline character

printf("Enter the key for the third encryption: ");

scanf("%d", &key);

getchar(); // Consume the newline character

railfence\_cipher(plaintext, key);

// Your name and roll number

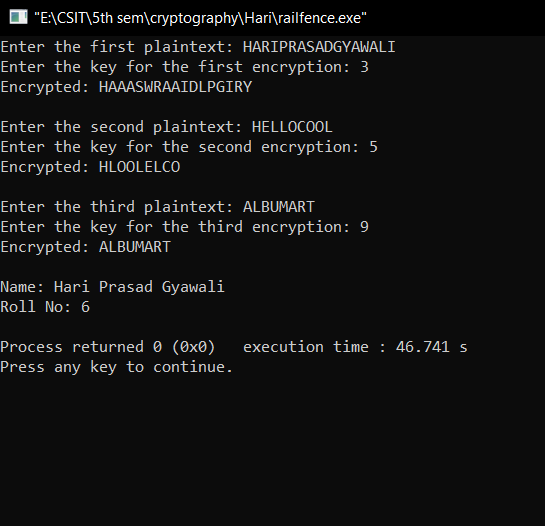
printf("\nName: Hari Prasad Gyawali\n");

printf("Roll No: 6\n");

return 0;

}

Output:



**Lab 06**

1. **Write a program to compute GCD of any two number**

#include <stdio.h>

// Function to calculate the GCD

int gcd(int a, int b) {

while (b != 0) {

int temp = b;

b = a % b;

a = temp;

}

return a;

}

int main() {

int num1, num2, num3;

// Input and calculation 1

printf("Enter the first integer: ");

scanf("%d", &num1);

printf("Enter the second integer: ");

scanf("%d", &num2);

// Calculate and print GCD

int result1 = gcd(num1, num2);

printf("GCD of %d and %d is: %d\n", num1, num2, result1);

// Input and calculation 2

printf("\nEnter the third integer: ");

scanf("%d", &num3);

// Calculate and print GCD

int result2 = gcd(result1, num3);

printf("GCD of previous result and %d is: %d\n", num3, result2);

// Input and calculation 3

int num4, num5;

printf("\nEnter another integer: ");

scanf("%d", &num4);

printf("Enter one more integer: ");

scanf("%d", &num5);

// Calculate and print GCD

int result3 = gcd(num4, num5);

printf("GCD of %d and %d is: %d\n", num4, num5, result3);

// Your name and roll number

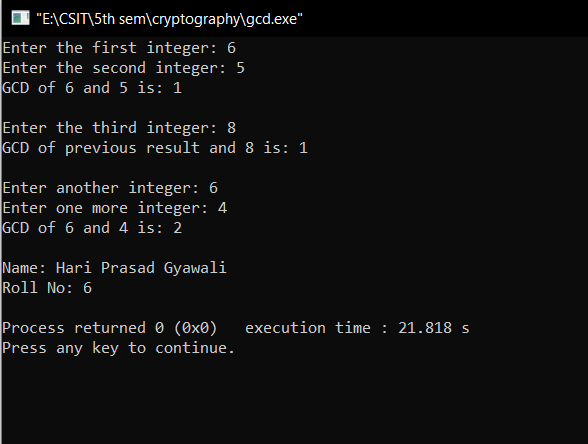
printf("\nName: Hari Prasad Gyawali\n");

printf("Roll No: 6\n");

return 0;

}

Output:



**Lab 07**

1. **Write a program to check wheather a number is multiplicative inverse of number using brute force method**

#include <stdio.h>

// Function to check if a is the multiplicative inverse of b

int isMultiplicativeInverse(int a, int b, int mod) {

int result = (a \* b) % mod;

return result == 1;

}

int main() {

int num1, num2;

// Input and check 1

printf("Enter the first integer: ");

scanf("%d", &num1);

printf("Enter the second integer: ");

scanf("%d", &num2);

if (isMultiplicativeInverse(num1, num2, 26)) {

printf("%d is the multiplicative inverse of %d.\n", num1, num2);

} else {

printf("%d is not the multiplicative inverse of %d.\n", num1, num2);

}

// Input and check 2

int num3, num4;

printf("\nEnter another integer: ");

scanf("%d", &num3);

printf("Enter one more integer: ");

scanf("%d", &num4);

if (isMultiplicativeInverse(num3, num4, 100)) {

printf("%d is the multiplicative inverse of %d.\n", num3, num4);

} else {

printf("%d is not the multiplicative inverse of %d.\n", num3, num4);

}

// Input and check 3

int num5, num6;

printf("\nEnter another integer: ");

scanf("%d", &num5);

printf("Enter one more integer: ");

scanf("%d", &num6);

if (isMultiplicativeInverse(num5, num6, 15)) {

printf("%d is the multiplicative inverse of %d.\n", num5, num6);

} else {

printf("%d is not the multiplicative inverse of %d.\n", num5, num6);

}

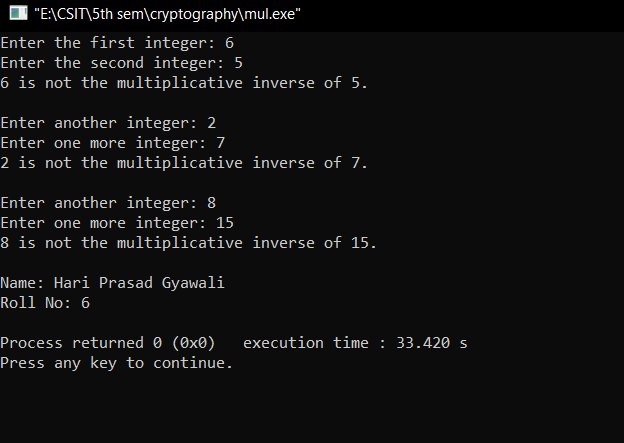
// Your name and roll number

printf("\nName: Hari Prasad Gyawali\n");

printf("Roll No: 6\n");

return 0;

}

Output:

**Lab 08**

1. **Write a program to compute totient of a number**

#include <stdio.h>

// Function to calculate the Totient of a number

int totient(int n) {

int result = 1; // Initialize result as n

for (int i = 2; i < n; i++) {

if (gcd(i, n) == 1) {

result++;

}

}

return result;

}

// Function to calculate the GCD

int gcd(int a, int b) {

while (b != 0) {

int temp = b;

b = a % b;

a = temp;

}

return a;

}

int main() {

int num1, num2, num3;

// Input and calculation 1

printf("Enter the first integer: ");

scanf("%d", &num1);

printf("Totient of %d is: %d\n", num1, totient(num1));

// Input and calculation 2

printf("\nEnter another integer: ");

scanf("%d", &num2);

printf("Totient of %d is: %d\n", num2, totient(num2));

// Input and calculation 3

printf("\nEnter one more integer: ");

scanf("%d", &num3);

printf("Totient of %d is: %d\n", num3, totient(num3));

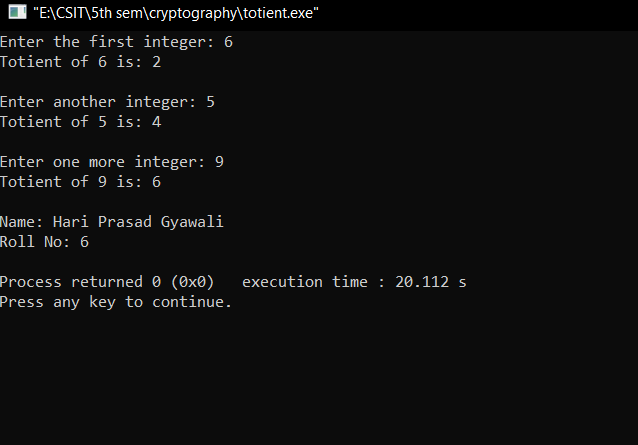
// Your name and roll number

printf("\nName: Hari Prasad Gyawali\n");

printf("Roll No: 6\n");

return 0;

}

Output:

**Lab 09**

1. **Write a program to compute multiplicative inverse of an integer**

#include <stdio.h>

// Function to calculate the GCD

int gcdExtended(int a, int b, int \*x, int \*y) {

// Base Case

if (a == 0) {

\*x = 0, \*y = 1;

return b;

}

int x1, y1;

int gcd = gcdExtended(b % a, a, &x1, &y1);

// Update x and y using results of recursive call

\*x = y1 - (b / a) \* x1;

\*y = x1;

return gcd;

}

// Function to calculate the multiplicative inverse

int modInverse(int a, int m) {

int x, y;

int g = gcdExtended(a, m, &x, &y);

if (g != 1) {

// Inverse doesn't exist

return -1;

} else {

// Making x positive

return (x % m + m) % m;

}

}

int main() {

int num1, num2, num3;

// Input and calculation 1

printf("Enter the integer: ");

scanf("%d", &num1);

int inverse1 = modInverse(num1, 26);

if (inverse1 == -1) {

printf("Multiplicative inverse does not exist for %d.\n", num1);

} else {

printf("Multiplicative inverse of %d (mod 26) is: %d\n", num1, inverse1);

}

// Input and calculation 2

printf("\nEnter another integer: ");

scanf("%d", &num2);

int inverse2 = modInverse(num2, 100);

if (inverse2 == -1) {

printf("Multiplicative inverse does not exist for %d.\n", num2);

} else {

printf("Multiplicative inverse of %d (mod 100) is: %d\n", num2, inverse2);

}

// Input and calculation 3

printf("\nEnter one more integer: ");

scanf("%d", &num3);

int inverse3 = modInverse(num3, 15);

if (inverse3 == -1) {

printf("Multiplicative inverse does not exist for %d.\n", num3);

} else {

printf("Multiplicative inverse of %d (mod 15) is: %d\n", num3, inverse3);

}

// Your name and roll number

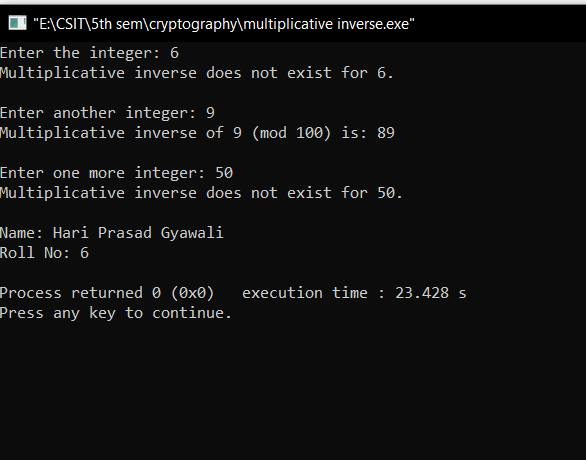
printf("\nName: Hari Prasad Gyawali\n");

printf("Roll No: 6\n");

return 0;

}

Output:



**Lab 10**

1. **Write a program to compute check whether two numbers are coprime or not**

#include <stdio.h>

// Function to calculate the GCD

int gcd(int a, int b) {

while (b != 0) {

int temp = b;

b = a % b;

a = temp;

}

return a;

}

// Function to check if two numbers are coprime

int areCoprime(int a, int b) {

return gcd(a, b) == 1;

}

int main() {

int num1, num2, num3;

// Input and check 1

printf("Enter the first integer: ");

scanf("%d", &num1);

printf("Enter the second integer: ");

scanf("%d", &num2);

if (areCoprime(num1, num2)) {

printf("%d and %d are coprime.\n", num1, num2);

} else {

printf("%d and %d are not coprime.\n", num1, num2);

}

// Input and check 2

int num4, num5;

printf("\nEnter another integer: ");

scanf("%d", &num4);

printf("Enter one more integer: ");

scanf("%d", &num5);

if (areCoprime(num4, num5)) {

printf("%d and %d are coprime.\n", num4, num5);

} else {

printf("%d and %d are not coprime.\n", num4, num5);

}

// Input and check 3

int num6, num7;

printf("\nEnter another integer: ");

scanf("%d", &num6);

printf("Enter one more integer: ");

scanf("%d", &num7);

if (areCoprime(num6, num7)) {

printf("%d and %d are coprime.\n", num6, num7);

} else {

printf("%d and %d are not coprime.\n", num6, num7);

}

// Your name and roll number

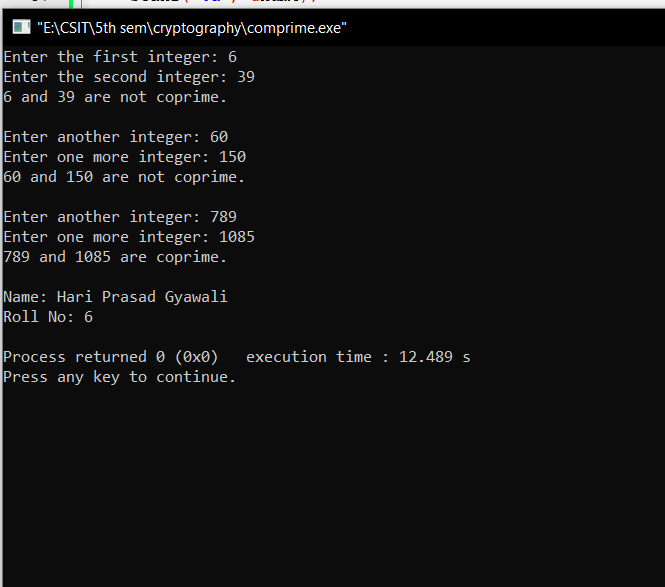
printf("\nName: Hari Prasad Gyawali\n");

printf("Roll No: 6\n");

return 0;

}

Output:



**Lab 11**

1. **Write a program to implement extended Euclidean Algorithm**

#include <stdio.h>

// Function to calculate the Extended Euclidean Algorithm

int extendedEuclidean(int a, int b, int \*x, int \*y) {

// Base Case

if (a == 0) {

\*x = 0, \*y = 1;

return b;

}

int x1, y1;

int gcd = extendedEuclidean(b % a, a, &x1, &y1);

// Update x and y using results of the recursive call

\*x = y1 - (b / a) \* x1;

\*y = x1;

return gcd;

}

int main() {

int num1, num2, num3, x, y;

// Input and calculation 1

printf("Enter the first integer: ");

scanf("%d", &num1);

printf("Enter the second integer: ");

scanf("%d", &num2);

// Calculate Extended Euclidean Algorithm

int gcd1 = extendedEuclidean(num1, num2, &x, &y);

printf("GCD of %d and %d is: %d\n", num1, num2, gcd1);

printf("Extended Euclidean Algorithm: %d \* %d + %d \* %d = %d\n", num1, x, num2, y, gcd1);

// Input and calculation 2

printf("\nEnter another integer: ");

scanf("%d", &num3);

// Calculate Extended Euclidean Algorithm

int gcd2 = extendedEuclidean(gcd1, num3, &x, &y);

printf("GCD of previous result and %d is: %d\n", num3, gcd2);

printf("Extended Euclidean Algorithm: %d \* %d + %d \* %d = %d\n", gcd1, x, num3, y, gcd2);

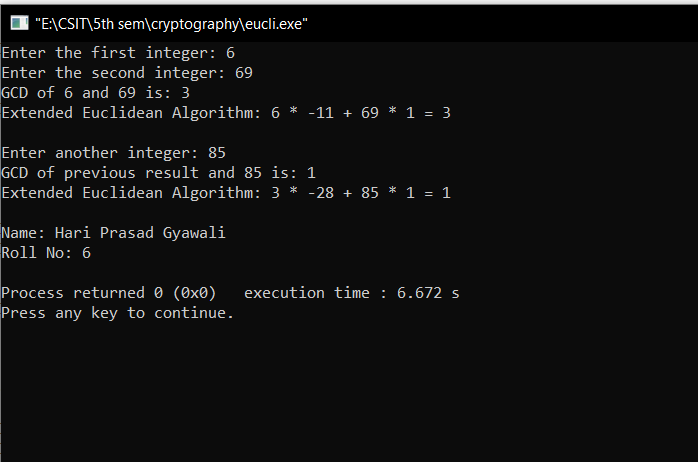
printf("\nName: Hari Prasad Gyawali\n");

printf("Roll No: 6\n");

return 0;

}

Output:



**Lab 12**

1. **Write a program to check whether given number is prime or not**

#include <stdio.h>

// Function to check if a number is prime

int isPrime(int num) {

if (num <= 1) {

return 0; // Not prime

}

for (int i = 2; i \* i <= num; i++) {

if (num % i == 0) {

return 0; // Not prime

}

}

return 1; // Prime

}

int main() {

int num1, num2, num3;

// Input and check 1

printf("Enter the first integer: ");

scanf("%d", &num1);

if (isPrime(num1)) {

printf("%d is prime.\n", num1);

} else {

printf("%d is not prime.\n", num1);

}

// Input and check 2

printf("\nEnter another integer: ");

scanf("%d", &num2);

if (isPrime(num2)) {

printf("%d is prime.\n", num2);

} else {

printf("%d is not prime.\n", num2);

}

// Input and check 3

printf("\nEnter one more integer: ");

scanf("%d", &num3);

if (isPrime(num3)) {

printf("%d is prime.\n", num3);

} else {

printf("%d is not prime.\n", num3);

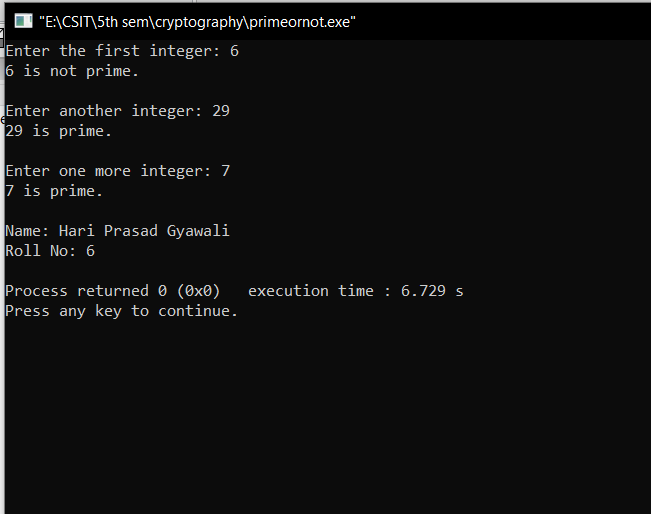
}

printf("\nName: Hari Prasad Gyawali\n");

printf("Roll No: 6\n");

return 0;

}

Output:  


**Lab 13**

1. **Write a program to perform primality checking using Rabin – Miller Algorithm**

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

long long mulmod(long long a, long long b, long long mod) {

long long x = 0, y = a % mod;

while (b > 0) {

if (b % 2 == 1) {

x = (x + y) % mod;

}

y = (y \* 2) % mod;

b /= 2;

}

return x % mod;

}

long long modulo(long long base, long long exponent, long long mod) {

long long x = 1;

long long y = base;

while (exponent > 0) {

if (exponent % 2 == 1)

x = (x \* y) % mod;

y = (y \* y) % mod;

exponent = exponent / 2;

}

return x % mod;

}

int Miller(long long p, int iteration) {

int i;

long long s;

if (p < 2) {

return 0;

}

if (p != 2 && p % 2 == 0) {

return 0;

}

s = p - 1;

while (s % 2 == 0) {

s /= 2;

}

for (i = 0; i < iteration; i++) {

long long a = rand() % (p - 1) + 1, temp = s;

long long mod = modulo(a, temp, p);

while (temp != p - 1 && mod != 1 && mod != p - 1) {

mod = mulmod(mod, mod, p);

temp \*= 2;

}

if (mod != p - 1 && temp % 2 == 0) {

return 0;

}

}

return 1;

}

int main() {

int iteration = 5;

long long num1, num2, num3;

// Input and check 1

printf("Enter the first integer to test primality: ");

scanf("%lld", &num1);

if (Miller(num1, iteration))

printf("\n%lld is definitely prime.\n", num1);

else

printf("\n%lld is not prime.\n", num1);

// Input and check 2

printf("\nEnter another integer to test primality: ");

scanf("%lld", &num2);

if (Miller(num2, iteration))

printf("\n%lld is definitely prime.\n", num2);

else

printf("\n%lld is not prime.\n", num2);

// Input and check 3

printf("\nEnter one more integer to test primality: ");

scanf("%lld", &num3);

if (Miller(num3, iteration))

printf("\n%lld is definitely prime.\n", num3);

else

printf("\n%lld is not prime.\n", num3);

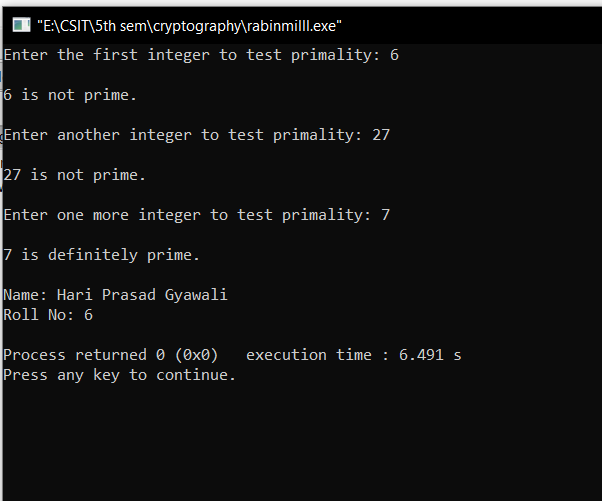
printf("\nName: Hari Prasad Gyawali\n");

printf("Roll No: 6\n");

return 0;

}

Output:



**Lab 14**

1. **Write a program to implement Diffie-Hellman algorithm**

#include <stdio.h>

#include <math.h>

// Function to calculate (base^exp) % mod

long long power(long long base, long long exp, long long mod) {

long long result = 1;

base = base % mod;

while (exp > 0) {

if (exp % 2 == 1) {

result = (result \* base) % mod;

}

base = (base \* base) % mod;

exp = exp / 2;

}

return result;

}

// Function to perform Diffie-Hellman key exchange

long long diffieHellman(long long base, long long private\_key, long long prime) {

return power(base, private\_key, prime);

}

int main() {

long long prime, base, private\_key\_A, private\_key\_B;

// Input for prime and base

printf("Enter the prime number (p): ");

scanf("%lld", &prime);

printf("Enter the base (g): ");

scanf("%lld", &base);

// Input for private key A

printf("Enter the private key for person A (a): ");

scanf("%lld", &private\_key\_A);

// Calculate public key A

long long public\_key\_A = diffieHellman(base, private\_key\_A, prime);

// Input for private key B

printf("Enter the private key for person B (b): ");

scanf("%lld", &private\_key\_B);

// Calculate public key B

long long public\_key\_B = diffieHellman(base, private\_key\_B, prime);

// Calculate shared secret

long long secret\_A = diffieHellman(public\_key\_B, private\_key\_A, prime);

long long secret\_B = diffieHellman(public\_key\_A, private\_key\_B, prime);

// Display results

printf("\nPublic Key A (g^a mod p): %lld\n", public\_key\_A);

printf("Public Key B (g^b mod p): %lld\n", public\_key\_B);

printf("\nShared Secret for A: %lld\n", secret\_A);

printf("Shared Secret for B: %lld\n", secret\_B);

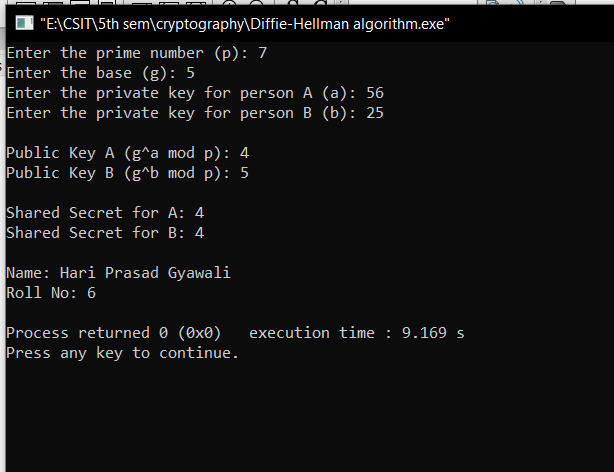
printf("\nName: Hari Prasad Gyawali\n");

printf("Roll No: 6\n");

return 0;

}

Output:



**Lab 15**

**15. Write a program perform key exchange and encryption-decryption using RSA algorithm**

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

long long gcd(long long a, long long b) {

if (b == 0)

return a;

return gcd(b, a % b);

}

long long power(long long base, long long exp, long long mod) {

long long result = 1;

base = base % mod;

while (exp > 0) {

if (exp % 2 == 1)

result = (result \* base) % mod;

base = (base \* base) % mod;

exp = exp / 2;

}

return result;

}

void generateKeys(long long p, long long q, long long\* n, long long\* e, long long\* d) {

\*n = p \* q;

long long phi = (p - 1) \* (q - 1);

for (\*e = 2; \*e < phi; (\*e)++) {

if (gcd(\*e, phi) == 1)

break;

}

for (\*d = 2; \*d < phi; (\*d)++) {

if ((\*d \* (\*e)) % phi == 1)

break;

}

}

long long encrypt(long long message, long long e, long long n) {

return power(message, e, n);

}

long long decrypt(long long encryptedMessage, long long d, long long n) {

return power(encryptedMessage, d, n);

}

int main() {

long long p, q, n, e, d;

for (int i = 0; i < 3; i++) {

// Input for prime numbers p and q

printf("\nEnter prime number p for set %d: ", i + 1);

scanf("%lld", &p);

printf("Enter prime number q for set %d: ", i + 1);

scanf("%lld", &q);

// Generate public and private keys

generateKeys(p, q, &n, &e, &d);

// Display public and private keys

printf("\nPublic Key (e, n) for set %d: (%lld, %lld)\n", i + 1, e, n);

printf("Private Key (d, n) for set %d: (%lld, %lld)\n", i + 1, d, n);

// Input for the message to be encrypted

long long message;

printf("\nEnter a message to encrypt for set %d: ", i + 1);

scanf("%lld", &message);

// Encrypt the message

long long encryptedMessage = encrypt(message, e, n);

printf("Encrypted Message for set %d: %lld\n", i + 1, encryptedMessage);

// Decrypt the message

long long decryptedMessage = decrypt(encryptedMessage, d, n);

printf("Decrypted Message for set %d: %lld\n", i + 1, decryptedMessage);

}

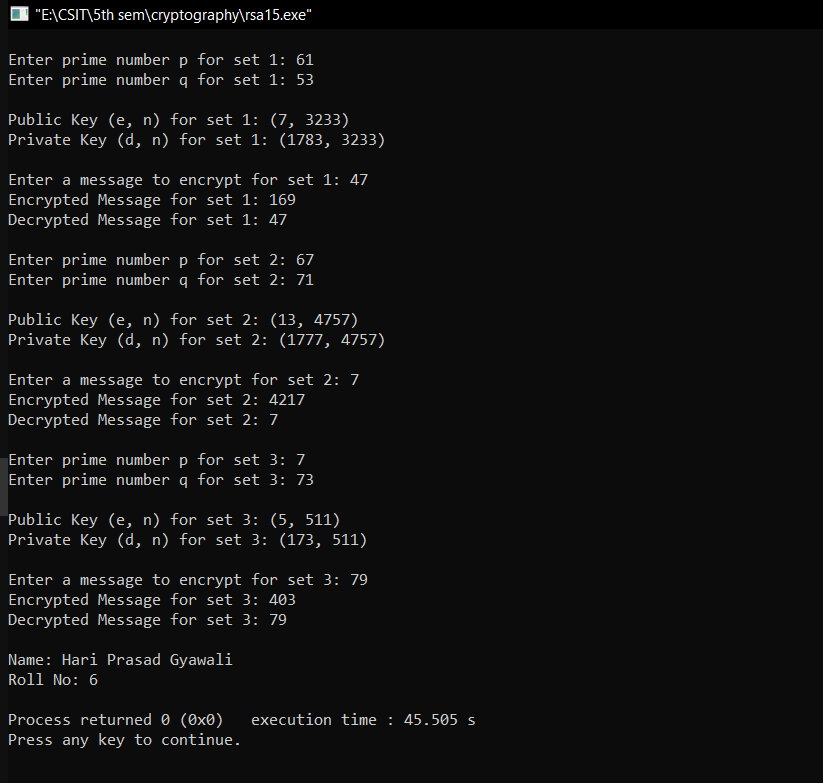
printf("\nName: Hari Prasad Gyawali\n");

printf("Roll No: 6\n");

return 0;

}

Output:

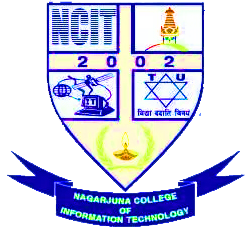


**LAB REPORT**

**OF**

**DESIGN AND ANALYSIS OF AGLORITHM**

**( CSC 314)**

****

**Submitted to**

**NAGARJUNA COLLEGE OF IT**

***(AFFILATED TO TRIBHUVAN UNIVERSITY)***

**Shankamul, Lalitpur- 09**

**Submitted By**

Hari Prasad Gyawali

***College Roll Number* : 06**

**Program : B.Sc. CSIT**

**Semester : IV**