

LAB REPORT ON CRYPTOGRAPHY

SUBMITTED BY : SAMYAK MANANDHAR SUBMITTED TO : SUMAN GUPTA

FACULTY : BSC.CSIT 5TH SEMESTER

**INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.N** | **TOPICS** | **DATE** | **SIGNATURE** |
| 1 | Write a program to implement Shift Cipher. | 9TH FEB 2025 |  |
| 2 | Write a program to implement Playfair Cipher. | 9TH FEB 2025 |  |
| 3 | Write a program to implement Rail Fence Cipher. | 2ND MAR 2025 |  |
| 4 | Write a program to implement Vigenere Cipher. | 2ND MAR 2025 |  |
| 5 | WAP to implement Euclidean Algorithm to find GCD of given numbers. | 2ND MAR 2025 |  |
| 6 | Write a program that computes additive inverse in given modulo n. | 2ND MAR 2025 |  |
| 7 | Write a program which takes two numbers and display whether they are relatively prime or not. | 9TH MAR 2025 |  |
| 8 | Write a program to implement Extended Euclidean Algorithm. | 9TH MAR 2025 |  |
| 9 | WAP to compute multiplicative inverse in given modulo n using Extended Euclidean Algorithm. | 9TH MAR 2025 |  |
| 10 | Write a program to implement Hill Cipher (Key matrix of size 2\*2/ Encryption/ Decryption. | 9TH MAR 2025 |  |
| 11 | WAP to demonstrate how output of S-Box (S1) is generated in DES. | 16TH MAR 2025 |  |
| 12 | Write a program to implement Robin Miller algorithm for primality test. | 16TH MAR 2025 |  |
| 13 | Write a program that takes any positive number and display the result after computing Totient value. | 16TH MAR 2025 |  |
| 14 | Write a program to compute primitive roots of given number. |  |  |
| 15 | WAP to compute discrete log of given number (provided the modulo and primitive root). |  |  |
| 16 | WAP to implement Diffie-Helman Key Exchange Algorithm. |  |  |
| 17 | WAP to implement RSA Algorithm (Encryption/Decryption). |  |  |
| 18 | WAP to implement Elgamal Cryptographic System. |  |  |
| 19 | Write a malicious logic code (Trojan Horse/Virus) program that performs some malicious works. |  |  |

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

**Algorithm for Caesar Cipher (Encrypting Letters & Digits)**

1. Start the program.
2. Display "Enter a message to encrypt".
3. Read the input string text.
4. Display "Enter the key".
5. Read the integer key.
6. Loop through each character in text:
7. If the character is a lowercase letter (a-z):

* Shift within 'a' to 'z' using: ch = ((ch - 'a' + key) % 26) + 'a'

1. Else if the character is an uppercase letter (A-Z):
   * Shift within 'A' to 'Z' using: ch = ((ch - 'A' + key) % 26) + 'A'
2. Else if the character is a digit (0-9):
   * Shift within '0' to '9' using: ch = ((ch - '0' + key) % 10) + '0'
3. Else (special character or space):
   * Leave it unchanged.
4. Output the encrypted message.
5. End the program.

**Code:**

#include <stdio.h>

#include <ctype.h>

#include <string.h>

int main() {

char text[500], ch;

int key;

printf("Enter a message to encrypt: ");

fgets(text, sizeof(text), stdin);

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

printf("Enter the key: ");

scanf("%d", &key);

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

ch = text[i];

if (isalnum(ch)) {

if (islower(ch)) {

ch = (ch - 'a' + key) % 26 + 'a';

}

if (isupper(ch)) {

ch = (ch - 'A' + key) % 26 + 'A';

}if (isdigit(ch)) {

ch = (ch - '0' + key) % 10 + '0';

}

}

text[i] = ch;

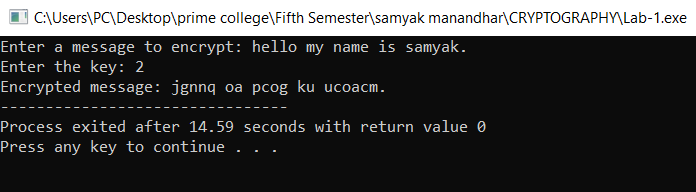
}

printf("Encrypted message: %s", text);

return 0;

}

**Output:**



**Lab 2: Write a program to implement Playfair Cipher.**

**Algorithm for Caesar Cipher Decryption (Letters & Digits)**

1. Start the program.
2. Display "Enter the text:".
3. Read the input string text.
4. Remove the newline character from text.
5. Display "Enter the key (shift value):".
6. Read the integer key.
7. Print the encrypted text.
8. Loop through each character in text:
   * If the character is a lowercase letter (a-z):
     + Decrypt using: ch = ((ch - 'a' - key + 26) % 26) + 'a'
   * Else if the character is an uppercase letter (A-Z):
     + Decrypt using: ch = ((ch - 'A' - key + 26) % 26) + 'A'
   * Else if the character is a digit (0-9):
     + Decrypt using: ch = ((ch - '0' - key + 10) % 10) + '0'
   * Else (special character or space):
     + Leave it unchanged.
9. Print the decrypted text.
10. End the program.

**Code:**

#include <stdio.h>

#include <ctype.h>

#include <string.h>

char decryptChar(char ch, int key) {

if (isalpha(ch)) {

char base = islower(ch) ? 'a' : 'A';

return (char)(((ch - base - key + 26) % 26) + base);

} else if (isdigit(ch)) {

return (char)(((ch - '0' - key + 10) % 10) + '0');

} else {

return ch;

}

}

void decrypt(char \*text, int key) {

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

text[i] = decryptChar(text[i], key);

}

}

int main() {

char text[500];

int key;

printf("Enter the text: ");

fgets(text, sizeof(text), stdin);

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

printf("Enter the key (shift value): ");

scanf("%d", &key);

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

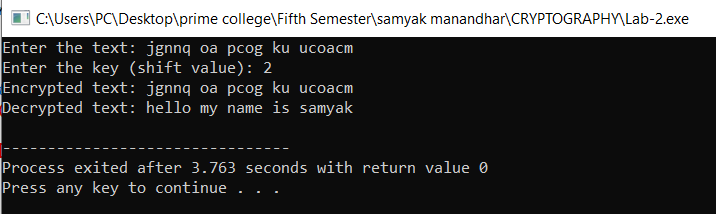
decrypt(text, key);

printf("Decrypted text: %s\n", text);

return 0;

}

**Output:**

****

**Lab 3: Write a program to implement Rail Fence Cipher.**

**Lab 4: Write a program to implement Vigenere Cipher.**

**Lab 5: WAP to implement Euclidean Algorithm to find GCD of given numbers.**

**Algorithm for Finding GCD (Greatest Common Divisor)**

Iterative Approach (Using Euclidean Algorithm)

1. Start the program.
2. Define a function gcdIterative(a, b):
   * While b is not 0:
     + Store b in a temporary variable.
     + Assign b = a % b.
     + Assign a = temp.
   * Return a as the GCD.
3. Define a function gcdRecursive(a, b):
   * If b == 0, return a as the GCD.
   * Else, recursively call gcdRecursive(b, a % b).
4. Start the program execution.
5. Prompt the user to enter two numbers.
6. Read the input numbers num1 and num2.
7. Compute the GCD using the iterative method and print the result.
8. Compute the GCD using the recursive method and print the result.
9. End the program.

**Code:**

#include<stdio.h>

int gcdIterative(int a, int b){

while(b != 0){

int temp = b;

b = a % b;

a = temp;

}

return a;

}

int gcdRecursive(int a, int b){

if (b==0)

return a;

return gcdRecursive(b, a % b);

}

int main(){

int num1, num2;

printf("enter two numbers:");

scanf("%d%d", &num1, &num2);

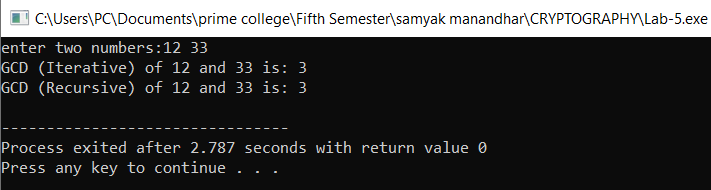
printf("GCD (Iterative) of %d and %d is: %d\n", num1, num2, gcdIterative(num1,num2));

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

return 0;

}

**Output:**



**Lab 6: Write a program that computes additive inverse in given modulo n.**

**Algorithm for Finding Additive Inverse Modulo n**

1. Start the program.
2. Define a function additiveInverse(a, n):
   * Compute the additive inverse using: inverse = (n - (a % n)) % n.
   * Return the computed inverse.
3. In the main function:
   * Prompt the user to enter a number a.
   * Read the input value a.
   * Prompt the user to enter a modulo n.
   * Read the input value n.
4. Check if n is greater than zero:
   * If n <= 0, print "Modulo n must be greater than zero." and exit the program.
5. Compute the additive inverse using additiveInverse(a, n).
6. Print the result:
   * Display "The additive inverse of a modulo n is: inverse".
7. End the program.

**Code:**

#include<stdio.h>

int additiveInverse(int a, int n){

int inverse = (n - (a % n)) % n;

return inverse;

}

int main(){

int a, n;

printf("Enter a number: ");

scanf("%d", &a);

printf("Enter a modulo n: ");

scanf("%d", &n);

if (n <= 0){

printf("Modulo n must be greater than zero.\n");

return 1;

}

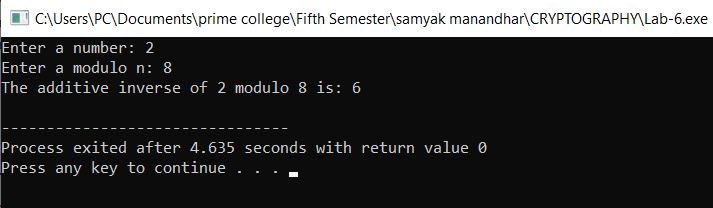
int result = additiveInverse(a, n);

printf("The additive inverse of %d modulo %d is: %d\n", a, n, result);

return 0;

}

**Output:**



**Lab 7: Write a program which takes two numbers and display whether they are relatively prime or not.**

**Algorithm to Check if Two Numbers are Relatively Prime**

1. Start
2. Input: Read two integers num1 and num2 from the user.
3. Define the function gcdIterative(a, b):
   * While b is not equal to 0:
     + Save the value of b in a temporary variable temp.
     + Set b to the remainder of a divided by b (a % b).
     + Set a to the value of temp.
   * Return the value of a (this will be the GCD of a and b).
4. Define the function are\_relatively\_prime(a, b):
   * Call the gcdIterative(a, b) function.
   * If the result of gcdIterative(a, b) is 1, return true (indicating a and b are relatively prime).
   * Else, return false (indicating a and b are not relatively prime).
5. Call the function are\_relatively\_prime(num1, num2) with the user inputs.
6. If are\_relatively\_prime(num1, num2) returns true:
   * Print that num1 and num2 are relatively prime.
7. Else:
   * Print that num1 and num2 are not relatively prime.
8. End

**Code:**

#include<stdio.h>

int gcdIterative(int a, int b){

while(b != 0){

int temp = b;

b = a % b;

a = temp;

}

return a;

}

int are\_relatively\_prime(int a, int b){

return gcdIterative(a,b) == 1;

}

int main(){

int num1, num2;

printf("enter first numbers:");

scanf("%d", &num1);

printf("enter second numbers:");

scanf("%d", &num2);

if (are\_relatively\_prime(num1, num2)){

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

}else{

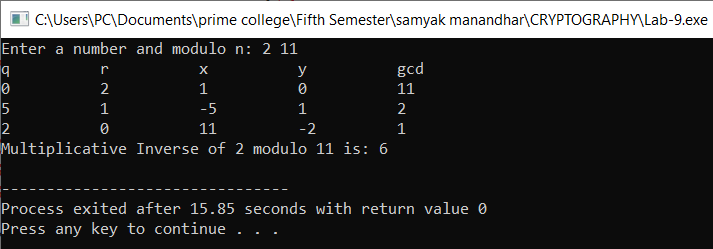
printf("%d and %d are not relatively prime.\n",num1, num2);

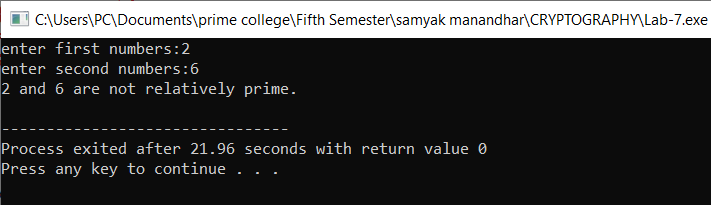
}

return 0;

}

**Output:**

****

****

**Lab 8: Write a program to implement Extended Euclidean Algorithm.**

**Algorithm to Find GCD and Coefficients Using Extended Euclidean Algorithm**

1. Start
2. Input: Read two integers num1 and num2 from the user.
3. Define the function extended\_gcd(a, b, x, y):
   * Initialize:
     + old\_x = 1, old\_y = 0 (These represent the coefficients for a initially).
     + current\_x = 0, current\_y = 1 (These represent the coefficients for b initially).
   * Print the table header: "q", "r", "x", "y", "gcd".
   * While b is not equal to 0:
     + Calculate q = a / b (quotient).
     + Calculate r = a % b (remainder).
     + Compute new coefficients x1 = old\_x - q \* current\_x and y1 = old\_y - q \* current\_y.
     + Print the current values of q, r, x1, y1, and b in a formatted table.
     + Update old\_x = current\_x, old\_y = current\_y, current\_x = x1, current\_y = y1.
     + Set a = b and b = r (move to the next iteration).
   * After exiting the loop:
     + Set \*x = old\_x and \*y = old\_y (store the coefficients in x and y).
     + Return the GCD (a), which is the last non-zero value of b.
4. Call the function extended\_gcd(num1, num2, &x, &y) with user inputs num1 and num2.
5. Output: Print the GCD of num1 and num2.
6. Output: Print the coefficients x and y that satisfy the equation num1 \* x + num2 \* y = GCD(num1, num2).
7. End

**Code:**#include<stdio.h>

int extended\_gcd(int a, int b, int \*x, int \*y){

int x1, y1;

int q, r;

int old\_x = 1, old\_y = 0;

int current\_x = 0, current\_y = 1;

printf("%-10s %-10s %-10s %-10s %-10s\n","q", "r", "x", "y", "gcd");

while (b !=0 ){

q = a / b;

r = a % b;

x1 = old\_x - q \* current\_x;

y1 = old\_y - q \* current\_y;

printf("%-10d %-10d %-10d %-10d %-10d\n",q ,r ,x1, y1, b);

old\_x = current\_x;

old\_y = current\_y;

current\_x = x1;

current\_y = y1;

a = b;

b = r;

}

\*x = old\_x;

\*y = old\_y;

return a;

}

int main(){

int num1, num2, x, y;

printf("Enter two numbers:");

scanf("%d%d",&num1, &num2);

int gcd = extended\_gcd(num1, num2, &x, &y);

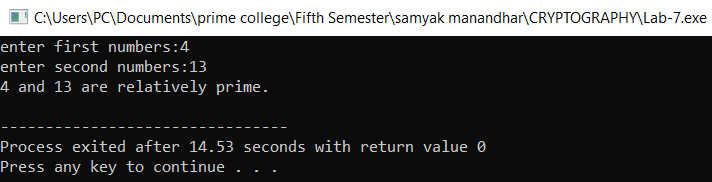
printf("\n GCD of %d and %d is %d\n", num1, num2, gcd);

printf("Coefficient: x= %d, y=%d1, y=%d\n", x, y);

return 0;

}

**Output:**

****

**Lab 9: WAP to compute multiplicative inverse in given modulo n using Extended Euclidean Algorithm.**

**Code:**

#include<stdio.h>

int extended\_gcd(int a, int b, int \*x, int \*y){

int x1, y1;

int q, r;

int old\_x = 1, old\_y = 0;

int current\_x = 0, current\_y = 1;

printf("%-10s %-10s %-10s %-10s %-10s\n","q", "r", "x", "y", "gcd");

while (b !=0 ){

q = a / b;

r = a % b;

x1 = old\_x - q \* current\_x;

y1 = old\_y - q \* current\_y;

printf("%-10d %-10d %-10d %-10d %-10d\n",q ,r ,x1, y1, b);

old\_x = current\_x;

old\_y = current\_y;

current\_x = x1;

current\_y = y1;

a = b;

b = r;

}

\*x = old\_x;

\*y = old\_y;

return a;

}

void mod\_inverse(int a, int n){

int x, y;

int gcd = extended\_gcd(a,n, &x, &y);

if (gcd != 1){

printf("\n Multiplicative inverse does not exist (GCD is not 1).\n");

}else {

int inverse = (x % n + n) % n;

printf("Multiplicative Inverse of %d modulo %d is: %d\n", a, n, inverse);

}

}

int main(){

int a, n;

printf("Enter a number and modulo n: ");

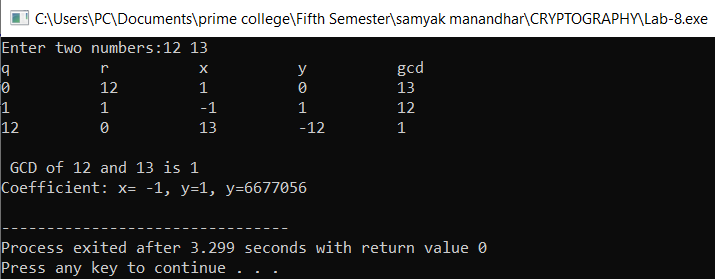
scanf("%d%d",&a, &n);

mod\_inverse(a, n);

return 0;

}

**Output:**

****

**Lab 10: Write a program to implement Hill Cipher (Key matrix of size 2\*2/ Encryption/ Decryption.**

**Code:**

**Output:**

**Lab 11: WAP to demonstrate how output of S-Box (S1) is generated in DES.**

**Code:**

**Output:**

**Lab 12: Write a program to implement Robin Miller algorithm for primality test.**

**Code:**

**Output:**

**Lab 13: Write a program that takes any positive number and display the result after computing Totient value**

**Code:**

**Output:**

**Lab 14: Write a program to compute primitive roots of given number.**

**Code:**

**Output:**

**Lab 15: WAP to compute discrete log of given number (provided the modulo and primitive root).**

**Code:**

**Output:**

**Lab 16: WAP to implement Diffie-Helman Key Exchange Algorithm.**

**Code:**

**Output:**

**Lab 17: WAP to implement RSA Algorithm (Encryption/Decryption).**

**Code:**

**Output:**

**Lab 18: WAP to implement Elgamal Cryptographic System.**

**Code:**

**Output:**

**Lab 19: Write a malicious logic code (Trojan Horse/Virus) program that performs some malicious works.**

**Code:**

**Output:**