22BCE3799

Apurba Koirala

Cryptography and Network Security Lab Assessment 2

1. Fermats theorem:

#include <iostream>

#include <cmath>

using namespace std;

bool prime(int p){

if(p<=1){

return false;

}

for (int i = 2; i <= sqrt(p); i++){

if(p%i == 0){

return false;

}

}

return true;

}

int gcd(int a, int b) {

if (b == 0) {

return a;

}

return gcd(b, a % b);

}

void fermats(int a, int b, int p){

if (b == p-1){

cout<<"Answer = "<<1;

return;

}

else if(b == p){

cout<<"Answer = "<<a;

return;

}

else{

cout<<a<<"^"<<b<<" MOD "<<p<<"\n";

int quotient = b / p;

int rem = b % p;

if (quotient+rem < p){

cout<<a<<"^"<<(quotient+rem)<<" MOD "<<p<<"\n";

double result = pow(a, b);

cout<<"Answer = "<< fmod(result, p)<<"\n";

}

else{

fermats(a, quotient+rem, p);

}

}

}

int main()

{

int a, b, p;

cout<< "enter a^b mod p values: ";

cin>>a;

cin>>b;

cin>>p;

if (!prime(p)){

cout<<"fermats theorem not applicable";

return 1;

}

else if(gcd(a, p) != 1){

cout<<"fermats theorem not applicable";

return 1;

}

else{

cout<<"conditions satisfied for fermats theorem.\n";

fermats(a, b, p);

}

return 0;

}

Output:

A black screen with white text

AI-generated content may be incorrect.

1. Euler’s Theorem to find remainder:

Code:

#include <iostream>

#include <cmath>

using namespace std;

bool isPrime(int num) {

if (num < 2) return false;

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

if (num % i == 0) return false;

}

return true;

}

int gcd(int a, int b) {

if (b == 0) return a;

return gcd(b, a % b);

}

int getPhi(int n) {

if (isPrime(n)) {

return n - 1;

}

for (int p = 2; p \* p <= n; p++) {

if (n % p == 0) {

int q = n / p;

if (isPrime(q)) {

cout << "p = " << p << "\tq = " << q << "\n";

return (p - 1) \* (q - 1);

}

}

}

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;

}

void eulers(int a, int b, int n) {

int phin = getPhi(n);

cout << "phi(n) = " << phin << "\n";

int rem = b % phin;

int result = pow(a, rem);

int answer = fmod(result, n);

cout << "Answer = " << answer << endl;

}

int main() {

int a, b, n;

cout << "Enter a, b, and n to compute a^b MOD n: ";

cin >> a >> b >> n;

if (gcd(a, n) != 1) {

cout << "Euler's theorem cannot be applied because gcd(a, n) != 1.\n";

return 1;

}

eulers(a, b, n);

return 0;

}

Output:   
Case 1:

When n is prime, phi(n) = n -1

A black background with white text

AI-generated content may be incorrect.

Case 2:

When n = p \* q, where p and q are prime numbers

A black screen with white text

AI-generated content may be incorrect.

Case 3: when n is neither a prime number, nor a product of two primes:

A black screen with white text

AI-generated content may be incorrect.

Extended Euclidean Algorithm:

Code:

#include <iostream>

#include <cmath>

using namespace std;

int extended\_euc(int r1, int r2, int s1, int s2, int t1, int t2){

int q = r1/r2;

int r = r1 % r2;

int s = s1 - s2 \* q;

int t = t1 - t2 \* q;

if (r == 0){

return r2;

}

return extended\_euc(r2, r, s2, s, r2, t);

}

int main(){

int r1, r2, s1, s2, t1, t2;

t2, s1 = 1;

s2, t1 = 0;

cout<<"Enter a and b for gcd(a, b): \n";

cin>> r1>> r2;

cout<<"gcd("<<r1<<", "<<r2<<")\n";

int answer = extended\_euc(r1, r2, s1, s2, t1, t2);

cout<<answer;

}

Output:

Same as class example:

A black screen with white text

AI-generated content may be incorrect.