Pre-Lab

1. What is Public-key cryptography, or asymmetric cryptography?

Public-key cryptography uses the same secret key for both encryption and decryption.

Asymmetric -key cryptography uses two different keys.

1. What is Symmetric key cryptography or Private key cryptography?

Symmetric key cryptography uses the same secret key for both encryption and decryption. Private key cryptography uses two different keys.

1. What is affine cipher?

Affine cipher is a type of substitution cipher that uses mathematical functions to encrypt letters.

1. What is Caesar cipher?

Caesar cipher shifts each letter in the plaintext by a fixed number of positions in the alphabet.

1. What is the Miller–Rabin primality test?

Miller-Rabin test is a algorithm to determine if a given number is likely prime.

1. What is the Euclidean Algorithm for finding GCD(A,B)?

Euclidean Algorithm repeatedly divides the larger number by the smaller one to find their greatest common divisor.

In-Lab

1. Affine Cipher

package Lab\_1;

import java.util.Scanner;

public class Affine {

// ax+b mod 26

public static String encrypt(String text, int a, int b) {

String result = "";

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

char c = text.charAt(i);

if (Character.isUpperCase(c)) {

result += (char)((a \* (c - 'A') + b) % 26 + 'A');

} else if (Character.isLowerCase(c)) {

result += (char)((a \* (c - 'a') + b) % 26 + 'a');

} else {

result += c;

}

}

return result;

}

// a\_inv(x-b) mod 26

public static String decrypt(String text, int a, int b) {

String result = "";

int a\_inv = 0;

for (int i = 0; i < 26; i++) {

if ((a \* i) % 26 == 1) {

a\_inv = i;

break;

}

}

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

char c = text.charAt(i);

if (Character.isUpperCase(c)) {

result += (char)((a\_inv \* (c - 'A' - b + 26) % 26) + 'A');

} else if (Character.isLowerCase(c)) {

result += (char)((a\_inv \* (c - 'a' - b + 26) % 26) + 'a');

} else {

result += c;

}

}

return result;

}

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

System.out.println("Enter the Text: ");

String t=sc.nextLine();

System.out.println("Enter the Value of a: ");

int a=sc.nextInt();

System.out.println("Enter the Value of b: ");

int b=sc.nextInt();

String enc=encrypt(t,a,b);

System.out.println("Encrypted Text:" + enc);

System.out.println("Decrypted Text:" + decrypt(enc,a,b));

}

}

1. Caesar Cipher

package Lab\_1;

import java.util.Scanner;

public class Caesar {

// x=c-a

// x+shift mod 26

public static String encrypt(String text, int shift) {

String result = "";

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

char c = text.charAt(i);

if (Character.isUpperCase(c)) {

result += (char)((c - 'A' + shift) % 26 + 'A');

} else if (Character.isLowerCase(c)) {

result += (char)((c - 'a' + shift) % 26 + 'a');

} else {

result += c;

}

}

return result;

}

// x=c+a

public static String decrypt(String text, int shift) {

return encrypt(text, 26 - shift);

}

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

System.out.println("Enter the Text: ");

String t=sc.nextLine();

System.out.println("Enter the Shift Value: ");

int shift=sc.nextInt();

String enc=encrypt(t,shift);

System.out.println("Encrypted Text:" + enc);

System.out.println("Decrypted Text:" + decrypt(enc,shift));

}

}

1. Miller-Rabin Primality Test

package Lab\_1;

import java.math.\*;

public class MillerRabin {

private static final int MAX\_ITERATIONS = 5;

public static boolean isPrime(BigInteger n) {

if (n.compareTo(BigInteger.ONE) <= 0) {

return false;

}

if (n.compareTo(BigInteger.valueOf(3)) <= 0) {

return true;

}

int s = 0;

BigInteger d = n.subtract(BigInteger.ONE);

while (d.mod(BigInteger.TWO).equals(BigInteger.ZERO)) {

s++;

d = d.divide(BigInteger.TWO);

}

for (int i = 0; i < MAX\_ITERATIONS; i++) {

BigInteger a = getRandomBase(n);

BigInteger x = a.modPow(d, n);

if (x.equals(BigInteger.ONE) || x.equals(n.subtract(BigInteger.ONE))) {

continue;

}

boolean isWitness = true;

for (int r = 1; r < s; r++) {

x = x.modPow(BigInteger.TWO, n);

if (x.equals(BigInteger.ONE)) {

return false;

}

if (x.equals(n.subtract(BigInteger.ONE))) {

isWitness = false;

break;

}

}

if (isWitness) {

return false;

}

}

return true;

}

private static BigInteger getRandomBase(BigInteger n) {

Random rand = new Random();

BigInteger result;

do {

result = new BigInteger(n.bitLength(), rand);

} while (result.compareTo(n) >= 0 || result.compareTo(BigInteger.ONE) <= 0);

return result;

}

}

1. Euclidean Algorithm for finding GCD(A,B)

package Lab\_1;

import java.util.Scanner;

public class Euclidean {

public static int gcd(int a, int b) {

if (b == 0) {

return a;

}

return gcd(b, a % b);

}

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

int a=sc.nextInt();

int b=sc.nextInt();

System.out.println(gcd(a, b));

}

}