**PASSWORD EXTRACTION:**

import hashlib

ascii\_hash = dict()

for i in range(-1,26):

for j in range(-1,26):

for k in range(26):

if (i==-1):

if (j==-1):

hash\_value = hashlib.md5(chr(65+k).encode())

ascii\_hash[chr(65+k)] = str(hash\_value.hexdigest())

continue

hash\_value = hashlib.md5((chr(65+j)+chr(65+k)).encode())

ascii\_hash[chr(65+j)+chr(65+k)] = str(hash\_value.hexdigest())

continue

hash\_value = hashlib.md5((chr(97+i)+chr(65+j)+chr(65+k)).encode())

ascii\_hash[chr(65+i)+chr(65+j)+chr(65+k)] = str(hash\_value.hexdigest())

key\_list = list(ascii\_hash.keys())

val\_list = list(ascii\_hash.values())

# print(val\_list)

while(True):

input\_text = input("\nEnter the text(-1 to exit): ")

if (input\_text=="-1"):

break

input\_text = input\_text.upper()

print(input\_text)

output\_hash = ascii\_hash[input\_text]

# print(ascii\_hash)

print("Output Hash: ",output\_hash)

print("Generating password from the hash...")

print("Password:",key\_list[val\_list.index(output\_hash)])

# for i in ascii\_hash:

# if (ascii\_hash[i] == output\_hash):

# print("Password: ",i)

**SHA:**

import java.util.Collections;

import java.util.Scanner;

import java.lang.String;

public class Main{

public static void leftshift(long val,int count)

{

String x = Long.toBinaryString(val);

if (x.length()<32) {

x = String.join("", Collections.nCopies(32-x.length(),"0"))+x;

}

x = x.substring(count)+x.substring(0,count);

String x1 = "";

for (int i=0;i<x.length();i=i+4) {

x1 += Integer.toHexString(Integer.parseInt(x.substring(i,i+4),2));

}

System.out.printf("\n shift of %h : %h \n",val,Long.parseLong(x1,16));

}

public static void main(String args[]){

long a = 0x67452301L;

long b = 0xabcdef12L;

long c = 0xa1b2c3e4L;

long d = 0x87654321L;

long e = 0x45ab67cdL;

long f = 0x12abcdefL;

int count;

Scanner ip= new Scanner(System.in);

System.out.print("enter the shift : ");

count= ip.nextInt();

leftshift(a,count);

leftshift(b,count);

leftshift(c,count);

leftshift(d,count);

leftshift(e,count);

leftshift(f,count);

long res = ((b&c)|((~b)&d));

System.out.printf("\n expression : %h \n",res);

long xor = a^b;

System.out.printf("\n xor(a and b) : %h \n",xor);

}

}

**PRIMITIVE ROOTS:**

import java.math.\*;

import java.util.\*;

public class primRoot {

public static void main(String arg[]){

Scanner scan = new Scanner(System.in);

System.out.print("Number:");

BigInteger p = new BigInteger(scan.nextLine());

BigInteger pi\_p = p.subtract(BigInteger.ONE);

BigInteger arr[] = new BigInteger[p.intValue()];

int iter = 0;

for (BigInteger i = new BigInteger("2");i.compareTo(pi\_p)<0;i=i.add(BigInteger.ONE)) {

if (i.isProbablePrime(1) && pi\_p.mod(i).compareTo(BigInteger.ZERO)==0) {

arr[iter++] = pi\_p.divide(i);

}

}

System.out.println("Working");

for (BigInteger i = new BigInteger("2");i.compareTo(p)<0;i=i.add(BigInteger.ONE)) {

int flag = 0;

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

if ((i.modPow(arr[j],p)).compareTo(new BigInteger("1"))==0) {

flag = 1;

break;

}

}

if (flag==0) {

System.out.print(i+", ");

}

}

}

}

**AES:**

import java.util.\*;

public class modAES{

static int key[][] = new int[4][44];

static int mainkey[][] = {

{0x2b, 0x28, 0xab, 0x9},

{0x7e, 0xae, 0xf7, 0xcf},

{0x15, 0xd2, 0x15, 0x4f},

{0x16, 0xa6, 0x88, 0x3c}

};

static final int[][] sbox = {

{0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30, 0x01, 0x67, 0x2b, 0xfe, 0xd7, 0xab, 0x76},

{0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0, 0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4, 0x72, 0xc0},

{0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc, 0x34, 0xa5, 0xe5, 0xf1, 0x71, 0xd8, 0x31, 0x15},

{0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a, 0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2, 0x75},

{0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0, 0x52, 0x3b, 0xd6, 0xb3, 0x29, 0xe3, 0x2f, 0x84},

{0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b, 0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c, 0x58, 0xcf},

{0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85, 0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c, 0x9f, 0xa8},

{0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5, 0xbc, 0xb6, 0xda, 0x21, 0x10, 0xff, 0xf3, 0xd2},

{0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17, 0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d, 0x19, 0x73},

{0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88, 0x46, 0xee, 0xb8, 0x14, 0xde, 0x5e, 0x0b, 0xdb},

{0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c, 0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95, 0xe4, 0x79},

{0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9, 0x6c, 0x56, 0xf4, 0xea, 0x65, 0x7a, 0xae, 0x08},

{0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6, 0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd, 0x8b, 0x8a},

{0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e, 0x61, 0x35, 0x57, 0xb9, 0x86, 0xc1, 0x1d, 0x9e},

{0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94, 0x9b, 0x1e, 0x87, 0xe9, 0xce, 0x55, 0x28, 0xdf},

{0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68, 0x41, 0x99, 0x2d, 0x0f, 0xb0, 0x54, 0xbb, 0x16}

};

static void display(){

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

for (int j=0; j < key[0].length; j++){

System.out.print(Integer.toHexString(key[i][j]) + " ");

}

System.out.println();

}

}

static void quadKey(int c){

int[] shift = new int[4];

int[] sboxer = new int[4];

int count=0;

int c1=0;

for (int i=0; i<4; i++){

String split;

if (i == 3){

split = Integer.toHexString(key[0][c-1]);

shift[count++] = key[0][c-1];

}

else{

split = Integer.toHexString(key[i+1][c-1]);

shift[count++]=key[i+1][c-1];

}

if (split.length() == 1){

split = "0" + split.charAt(0);

}

int row = Integer.parseInt(split.charAt(0) + "", 16);

int col = Integer.parseInt(split.charAt(1) + "", 16);

sboxer[c1++]=sbox[row][col];

}

System.out.println("Top circular shift: ");

for (int n=0;n<4;n++){

System.out.println(Integer.toHexString(shift[n])+"\t");

}

System.out.println("\n");

System.out.println("Substitute bytes: ");

for (int m=0;m<4;m++){

System.out.println(Integer.toHexString(sboxer[m])+"\t");

}

}

public static void main(String[] args){

Scanner sc = new Scanner(System.in);

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

for (int j=0; j<mainkey[0].length; j++){

key[i][j] = mainkey[i][j];

}

}

System.out.println("Main Key: ");

display();

int count = 1;

quadKey(4);

}

}