# **Diffie-Hellman Key Exchange**

**Program**

#include<stdio.h>

#include<conio.h>

#include<math.h>

#include<stdlib.h>

#include<time.h>

int prime(int number);

int primitiveroot(int a, int p);

int main()

{

srand(time(NULL));

int Xa, Xb;

long long int number;

int KeyA,KeyB;

int Ya, Yb, p, g;

do{

p=(rand()%100) + 3;

}while (prime(p));

printf("p=%d\n", p);

g=1;

while (1){

g=(rand()%(p-1))+1;

if (primitiveroot(g, p)){

break;

}

}

printf("g=%d\n", g);

printf("Enter the value of Xa and Xb(less than %d)\n", p);

scanf("%d%d", &Xa, &Xb);

Ya=pow(g, Xa);

Ya=Ya% p;

number=pow(g, Xb);

Yb=(int)(number % p);

printf("\nYA=%d\n", Ya);

printf("YB=%d\n", Yb);

KeyA=pow(Yb, Xa);

KeyA=KeyA%p;

number=pow(Ya, Xb);

KeyB=number%p;

printf("\nKEYA=%d\n", KeyA);

printf("KEYB=%d\n", KeyB);

if (KeyA==KeyB){

puts("Same");

}else{

puts("not same");

}

getch();

}

int prime(int number)

{

int flag=0;

for (int i=2; i<(number/2); i++){

if (number%i==0){

flag=1;

break;

}

}

if (flag){

return 1;

}else{

return 0;

}

}

int primitiveroot(int a, int p)

{

int array[30], j;

j=0;

for (int i=1; i<p; i++){

array[j]=pow(a, i);

array[j]=array[j] % p;

// printf("array[%d]=%d\n", j, array[j]);

j++;

}

int flag=0;

for (int i=0; i<(p-1); i++){

for (int k=k+1; k<p;k++){

if (array[i]==array[k]){

flag=1;

break;

}

}

}

if (flag==0){

return 0;

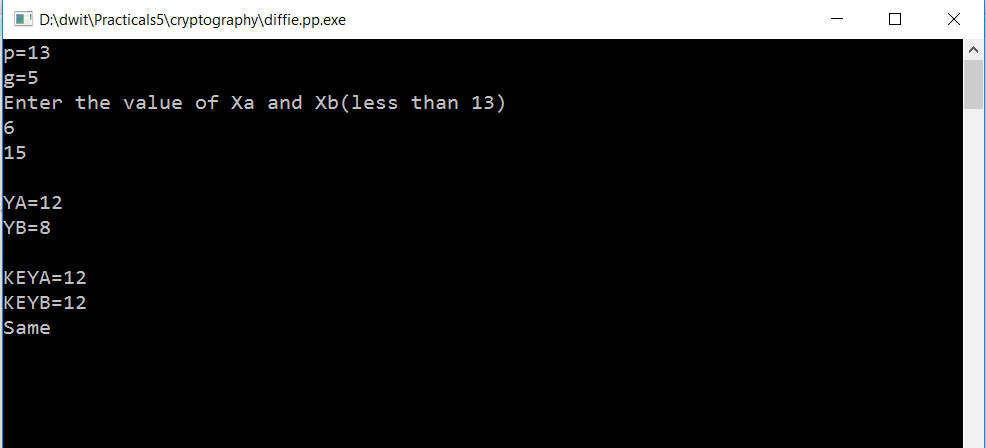
}else{

return 1;

}

}

**Output**



# **Lucas Lehmer Test**

**Program**

#include<stdio.h>

#include<math.h>

void lucas\_function(int p);

int main()

{

int p;

puts("Enter a number");

scanf("%d", &p);

lucas\_function(p);

}

void lucas\_function(int p)

{

int s=4;

int m=pow(2, p) -1;

// printf("%d", m);

for (int i=0; i<(p-2);i++){

s=((s\*s)-2)%m;

}

if (s==0){

puts("PRIME.");

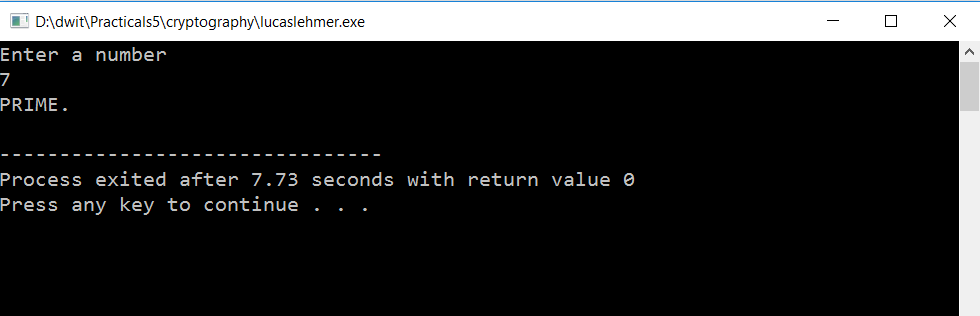
}else{

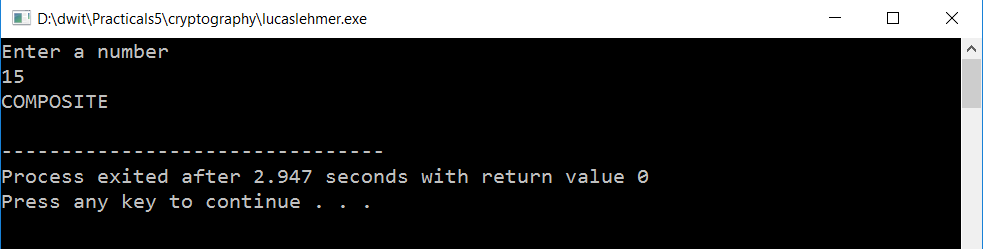
puts("COMPOSITE");

}

}

**Output**





# **Primitive Root of a Prime Number**

**Program**

#include<stdio.h>

#include<math.h>

int main()

{

int a, p, array[30], j;

puts("Enter value(a)");

scanf("%d", &a);

puts("Enter value(p)");

scanf("%d", &p);

j=0;

for (int i=1; i<p; i++){

array[j]=pow(a, i);

array[j]=array[j] % p;

printf("array[%d]=%d\n", j, array[j]);

j++;

}

int flag=0;

for (int i=0; i<(p-1); i++){

for (int k=k+1; k<p;k++){

if (array[i]==array[k]){

flag=1;

break;

}

}

}

if (flag==0){

printf("%d is not primitive root of %d", a, p);

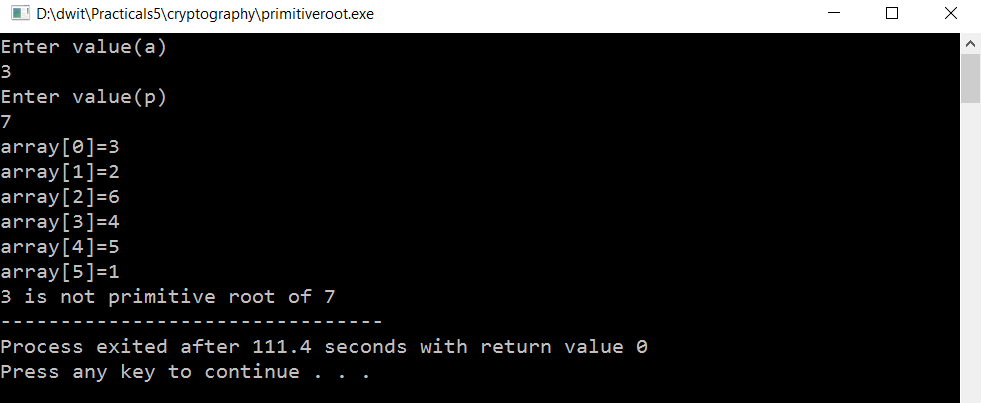
}else{

printf("%d is primitive root of %d", a, p);

}

}

**Output**



# **RSA(Rivest Shamir Adleman)**

**Program**

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

#include<math.h>

int gcd(int n , int i);

int encryption(int m, int e, int n);

int decryption(int c, int d, int n);

int main()

{

srand(time(NULL));

int p, q, n, flag=1, e, m=2, d;

int totient;

puts("Enter the value of p(large prime number)");

scanf("%d", &p);

puts("Enter the value of q(large prime number)");

scanf("%d", &q);

n=p\*q;

totient=(p-1)\*(q-1);

while (flag){

e=(rand()%totient)+1;

if (gcd(e, totient)==1){

flag=0;

}

}

// printf("m=%d, e=%d, n=%d", m, e, n);

for (int x=0; x<1000; x++){

if ((e\*x)%totient==1){

d=x;

break;

}

}

decryption(encryption(m, e, n), d,n);

}

int gcd(int n , int i)

{

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

if (n%j==0 && i%j==0){

return j;

}

}

return 1;

}

int decryption(int c, int d, int n)

{

int message=pow(c, d);

message=message%n;

printf("\nDecryption=%d", message);

}

int encryption(int m, int e, int n)

{

int c;

c=pow(m, e);

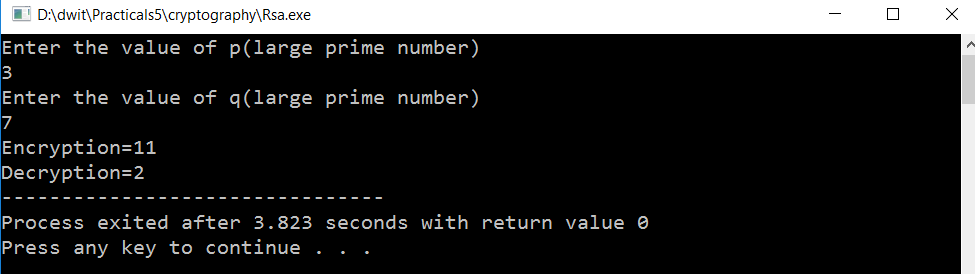
c=c%n;

printf("Encryption=%d", c);

return c;

}

**Output**



# **Euler’s Totient**

**Program**

#include<stdio.h>

#include<conio.h>

int gcd(int n , int i);

int main()

{

int n, count=0;

puts("ENTER a number(n)");

scanf("%d", &n);

for (int i=1; i<n; i++){

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

count++;

}

}

printf("Euler's Totient(%d) : %d",n, count);

}

int gcd(int n , int i)

{

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

if (n%j==0 && i%j==0){

return j;

}

}

return 1;

}

**Output**

