**Lab 2**

**Multiplicative cipher**

**Code:**

#include<bits/stdc++.h>

using namespace std;

int multiplicativeInverse(int a,int n)

{

int q,r1,r2,r,t1,t2,t;

r1=n;

r2=a;

t1=0;

t2=1;

while(r2>0)

{

q=r1/r2;

r=r1-(q\*r2);

r1=r2;

r2=r;

t=t1-(q\*t2);

t1=t2;

t2=t;

}

if(r1!=1)

{

return 0;

}

else

{

if(t1<0)

{

while(t1<0)

{

t1+=n;

if(t1>=0)

{

break;

}

}

}

}

return t1;

}

string encrypt(string text,int k)

{

string cipher ="";

int i;

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

{

if(islower(text[i]))

{

cipher += char(int( (text[i]-97)\*k)%26 + 97);

}

else

{

cipher += char(int(( text[i]-65)\*k)%26 + 65);

}

}

return cipher;

}

string decrypt(string cipher,int k)

{

//decryption

string decryptedText="";

int i,temp;

for(i=0; i<cipher.length(); i++)

{

if(islower(cipher[i]))

{

temp=(cipher[i]-97)\*k;

if(temp<0)

{

temp+=26;

}

decryptedText += char(int(temp)%26 + 97);

}

else

{

temp=(cipher[i]-65)\*k;

while(temp<0)

{

temp+=26;

}

decryptedText += char(int(temp)%26 + 65);

}

}

return decryptedText;

}

int main()

{

cout << "Enter Plain Text : ";

string text;

cin >> text;

cout << "Enter Key : ";

int k,k1,i;

cin >> k;

string cipher=encrypt(text,k);

int x;

x=multiplicativeInverse(k,26);

string decryptedText=decrypt(cipher,x);

cout << endl;

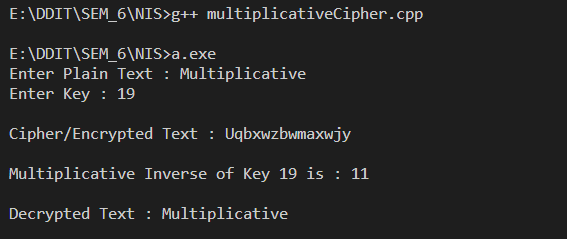
cout << "Cipher/Encrypted Text : " << cipher << endl;

cout << "\nMultiplicative Inverse of Key " << k << " is : " << x << endl;

cout << "\nDecrypted Text : " << decryptedText << endl;

return 0;

}



**Multiplicative Inverse:**

**Code:**

//find a (multiplicative inverse of n) using extended euclidean algorithm

#include<bits/stdc++.h>

using namespace std;

#define ll long long

ll multiplicativeInverse(ll a,ll n)

{

ll q,r1,r2,r,t1,t2,t;

r1=n;

r2=a;

t1=0;

t2=1;

while(r2>0)

{

q=r1/r2;

r=r1-(q\*r2);

r1=r2;

r2=r;

t=t1-(q\*t2);

t1=t2;

t2=t;

}

if(r1!=1)

{

return 0;

}

else

{

if(t1<0)

{

while(t1<0)

{

t1+=n;

if(t1>=0)

{

break;

}

}

}

}

return t1;

}

int main()

{

ll a,n;

cout << "Enter a : " ;

cin >> a;

cout << "Enter n : ";

cin >> n;

ll x;

x=multiplicativeInverse(a,n);

if(x==0)

{

cout << "\nMultiplicative inverse of " << a << " does not exist." << endl;

}

else

{

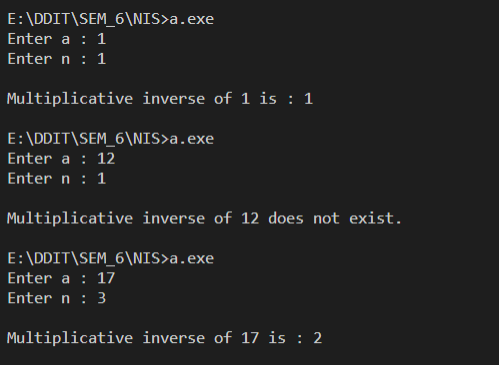
cout << "\nMultiplicative inverse of " << a << " is : " << x << endl;

}

return 0;

}

Example



**Affine Cipher:**

**Code:**

#include<bits/stdc++.h>

using namespace std;

#define ll long

ll multiplicativeInverse(ll a,ll n)

{

ll q,r1,r2,r,t1,t2,t;

r1=n;

r2=a;

t1=0;

t2=1;

while(r2>0)

{

q=r1/r2;

r=r1-(q\*r2);

r1=r2;

r2=r;

t=t1-(q\*t2);

t1=t2;

t2=t;

}

if(r1!=1)

{

return 0;

}

else

{

if(t1<0)

{

while(t1<0)

{

t1+=n;

if(t1>=0)

{

break;

}

}

}

}

return t1;

}

string encrypt(string text,ll k1,ll k2)

{

//encryption

string cipher ="";

int i;

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

{

//c=(P\*K1+K2)%26

if(islower(text[i]))

{

cipher += char(int( ((text[i]-97)\*k1)+k2 )%26 + 97);

}

else

{

cipher += char(int( (( text[i]-65)\*k1)+k2 )%26 + 65);

}

}

return cipher;

}

string decrypt(string cipher,ll k2,ll x)

{

string decryptedText="";

int i,temp;

for(i=0; i<cipher.length(); i++)

{

if(islower(cipher[i]))

{

temp=((cipher[i]-97)-k2)\*x;

while(temp<0)

{

temp+=26;

if(temp>=0)

break;

}

decryptedText += char(int(temp)%26 + 97);

}

else

{

temp=((cipher[i]-65)-k2)\*x;

while(temp<0)

{

temp+=26;

if(temp>=0)

break;

}

decryptedText += char(int(temp)%26 + 65);

}

}

return decryptedText;

}

int main()

{

cout << "Enter Plain Text : " ;

string text;

cin >> text;

cout << "Enter Key K(K1,K2) : ";

ll k1,k2;

cin >> k1 >> k2;

int x;

x=multiplicativeInverse(k1,26);

string cipher=encrypt(text,k1,k2);

string decryptedText=decrypt(cipher,k2,x);

cout << "\nCipher/Encrypted Text : " << cipher << endl;

cout << "\nMultiplicative Inverse of Key " << k1 << " is : " << x << endl;

cout << "\nDecrypted Text : " << decryptedText << endl;

return 0;

}

Example:

