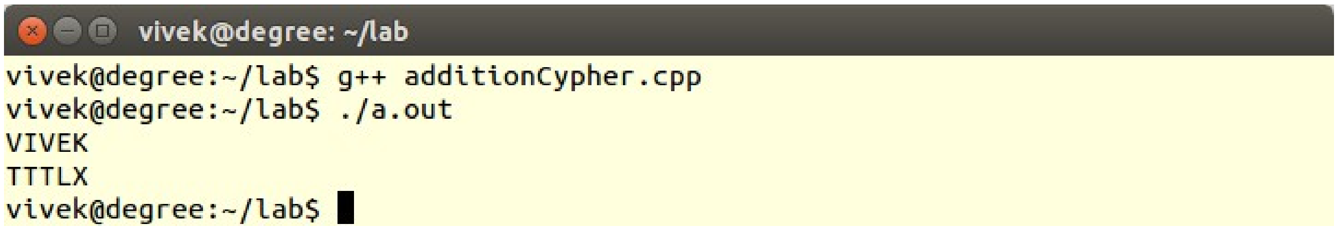
1.Additional Cipher

```
#include<bits/stdc++.h>
int encrypt(int p)
{
	return (2*p+3)%26;
}
using namespace std;
int main()
{
	char c[101];
	char p[101];
	cin>>p;
	int i=0;
	for(i=0;p[i]!='\0';i++ ) {
		c[i]=encrypt(p[i]-'A');
		c[i]+='A';
	}
	c[i]='\0';
	cout<<c<<endl;
}
```

input-output:

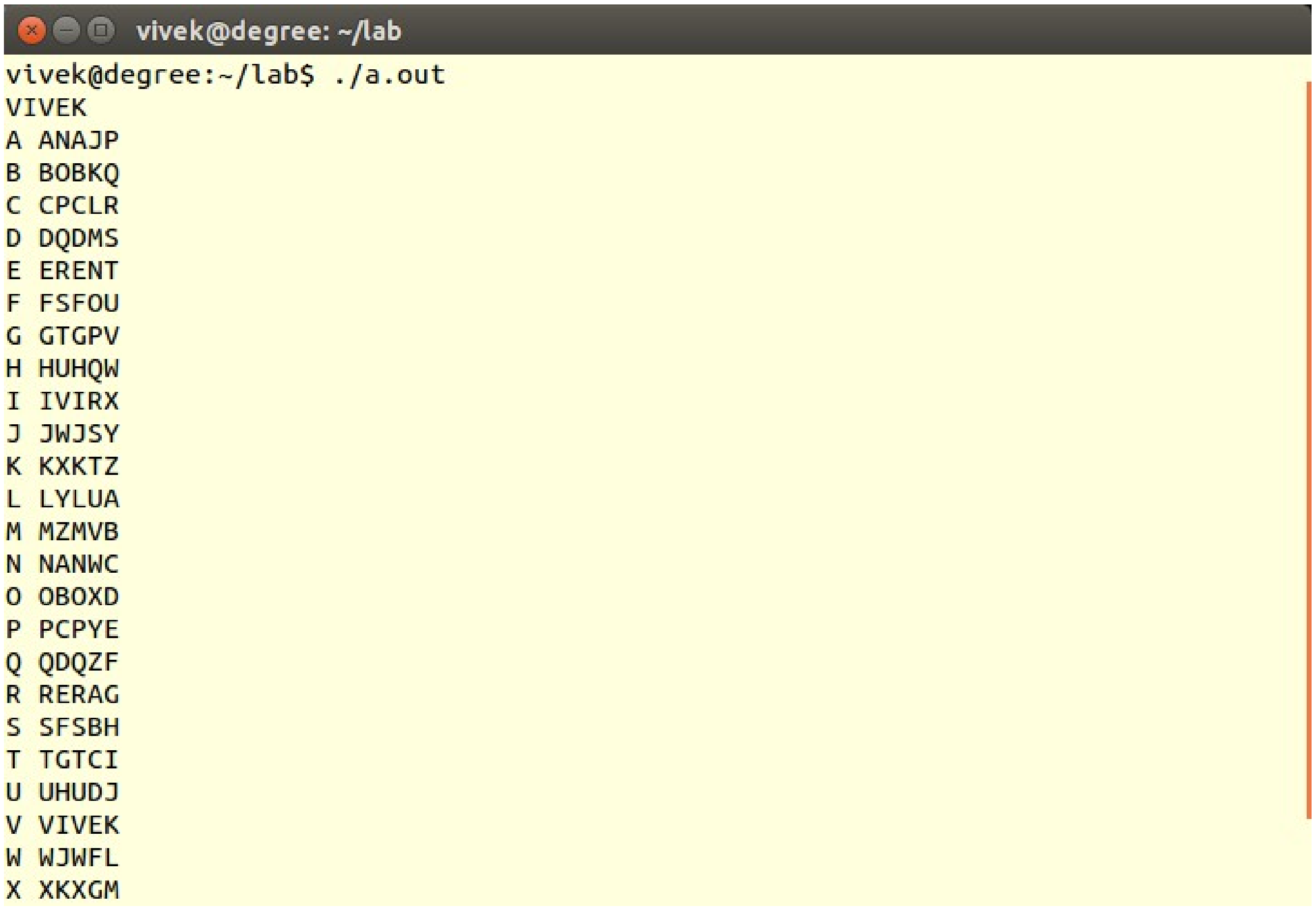


```
vivek@degree:~/lab$ g++ additionCypher.cpp
vivek@degree:~/lab$ ./a.out
VIVEK
TTTLX
vivek@degree:~/lab$
```

2.Brute Force Attack On Additive Cipher

```
#include<bits/stdc++.h>
using namespace std;
int encrypt(int p,int b)
{
        return ((p-b+26)%26);
}
int main()
{
        char c[101],p[101];
        cin>>p;
        int i=0, b,n=strlen(p);
        for(i=0;i<n;i++ )  hash[p[i]-'A']++;
        int maxindex=0,max=0;
        for(int i=0;i<26;i++ ) if(hash[i]>max) max=hash[i]; maxindex=i;
        for(int J=0;J<26;J++) {
                b=(maxindex-J)%26;
                for(i=0;i<n;i++ ) c[i]=encrypt(p[i]-'A',b)+'A';
                c[i]='\0';
                cout<<(char)(J+'A')<<" "<<c<<endl;  }

}
```

input-output:

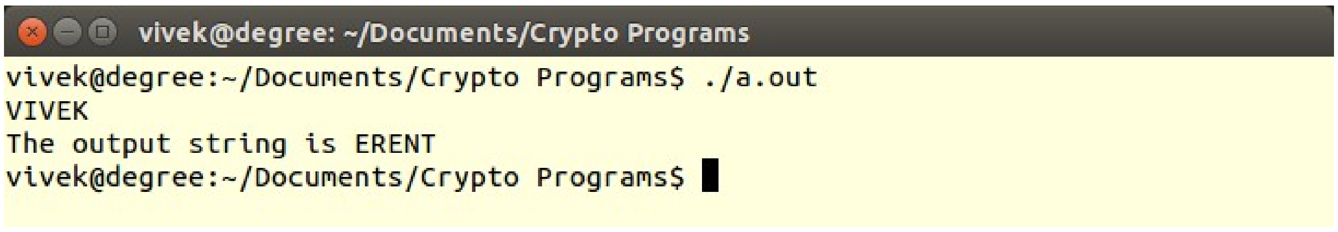


```
vivek@degree: ~/lab
vivek@degree:~/lab$ ./a.out
VIVEK
A ANAJP
B BOBKQ
C CPCLR
D DQDMS
E ERENT
F FSFOU
G GTGPV
H HUHQW
I IVIRX
J JWJSY
K KXKTZ
L LYLUA
M MZMVB
N NANWC
O OBOXD
P PCPYE
Q QDQZF
R RERAG
S SFSBH
T TGTCI
U UHUDJ
V VIVEK
W WJWFL
X XKXGM
```

3.Statical Attack on Additive cipher

```
#include <iostream>
#include <limits.h>

using namespace std;
int main() {
        string str,strOutput;
        int hash[26] = {0};
       int i;
        for(i = 0; i < str.size(); i++) hash[str[i] - 'A']++;
        int maxIndex = INT_MIN, maxValue = INT_MIN;
        for(i = 0; i < 26; i++){
            if(maxValue < hash[i]) {
                        maxValue = hash[i];
                        maxIndex = i;
                }
       }
        int b = maxIndex - 4;
        if(b < 0) b += 26;
        for(i = 0; i < str.size(); i++) {
                char ch = (str[i] - 'A' - b) % 26 + 'A';
                if(ch < 'A') ch += 26;
                 strOutput.push_back(ch);
        }
        cout << "The output string is " << strOutput<<endl;
      return 0;
}
```

input-output:



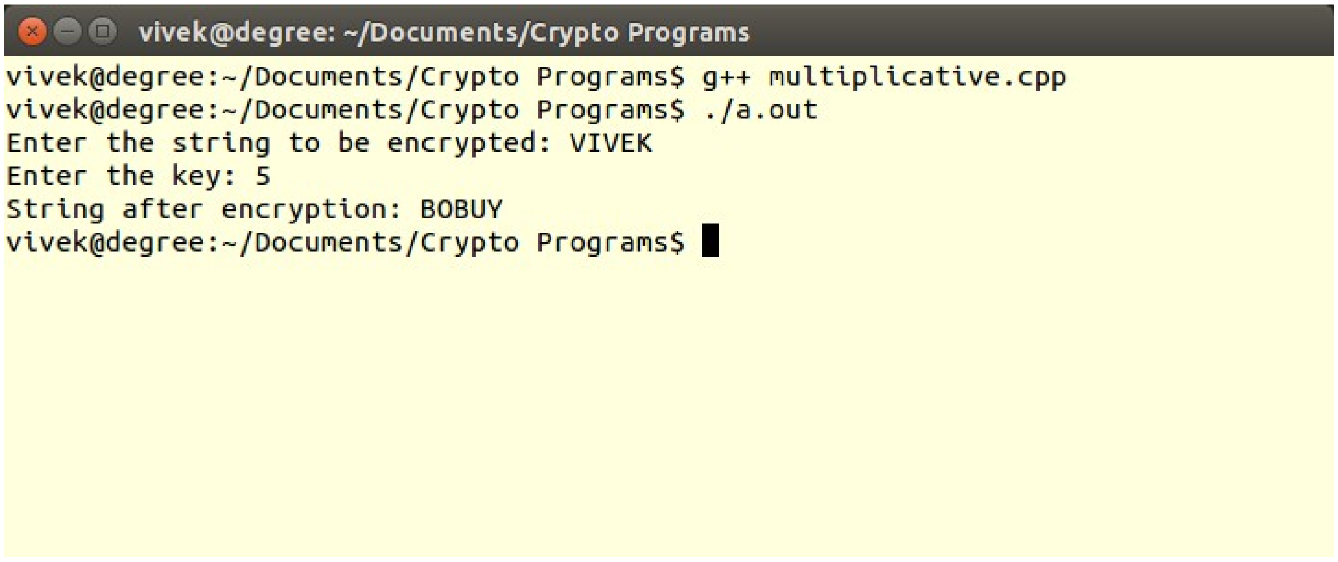
vivek@degree: ~/Documents/Crypto Programs

```
vivek@degree:~/Documents/Crypto Programs$ ./a.out
VIVEK
The output string is ERENT
vivek@degree:~/Documents/Crypto Programs$
```

4.Multiplicative Cipher

```
#include <iostream>
using namespace std;
int main() {
	string str;
	cin >> str;
	cout << "Enter the key: ";
	int b;
	cin >> b;
	if(b % 13 == 0 || b % 2 == 0) {
		cout << "Not a valid Key";
	} else {
		for(int i = 0; i < str.size(); i++) {
			str[i] = ((str[i] - 'A') * b) % 26 + 'A';
		}
		cout << "Ciphertext " << str<<endl;
	}
	return 0;
}
```

input-output:

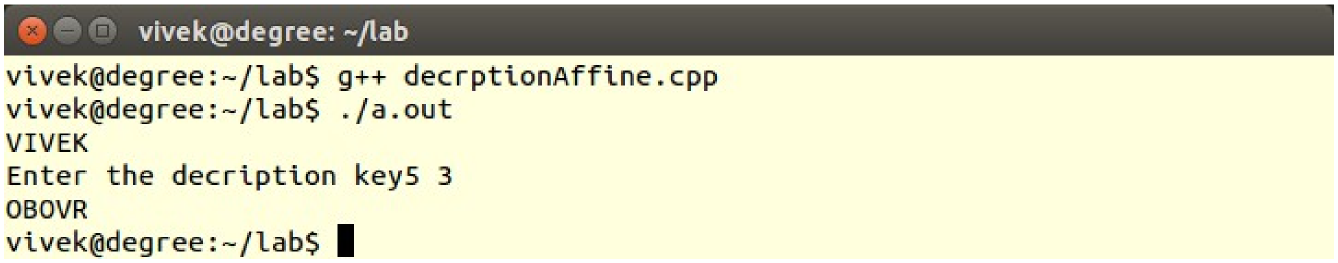


```
vivek@degree: ~/Documents/Crypto Programs
vivek@degree:~/Documents/Crypto Programs$ g++ multiplicative.cpp
vivek@degree:~/Documents/Crypto Programs$ ./a.out
Enter the string to be encrypted: VIVEK
Enter the key: 5
String after encryption: BOBUY
vivek@degree:~/Documents/Crypto Programs$ 
```

5.Affine Cipher

```
#include<bits/stdc++.h>
int encrypt(int p,int ainv,int b)
{
        return (((ainv*(p-b+26)))%26);
}
using namespace std;
int hash[26];
int main()
{

        char c[101] , p[101];
        cin>>c;
        int a,b;
        cout<<"Enter the decription key";
        cin>>a>>b;
        int ainv;
        for(int i=0;i<26;i++) {
                 if((i*a)%26==1) {
                        ainv=i;
                        break;
                }
        }
        int i=0, n=strlen(c);
        for(i=0;i<n;i++ )
                        p[i]=encrypt(c[i]-'A',ainv,b)+'A';
        p[i]='\0';
        cout<<p<<endl;
}
```

input-output:

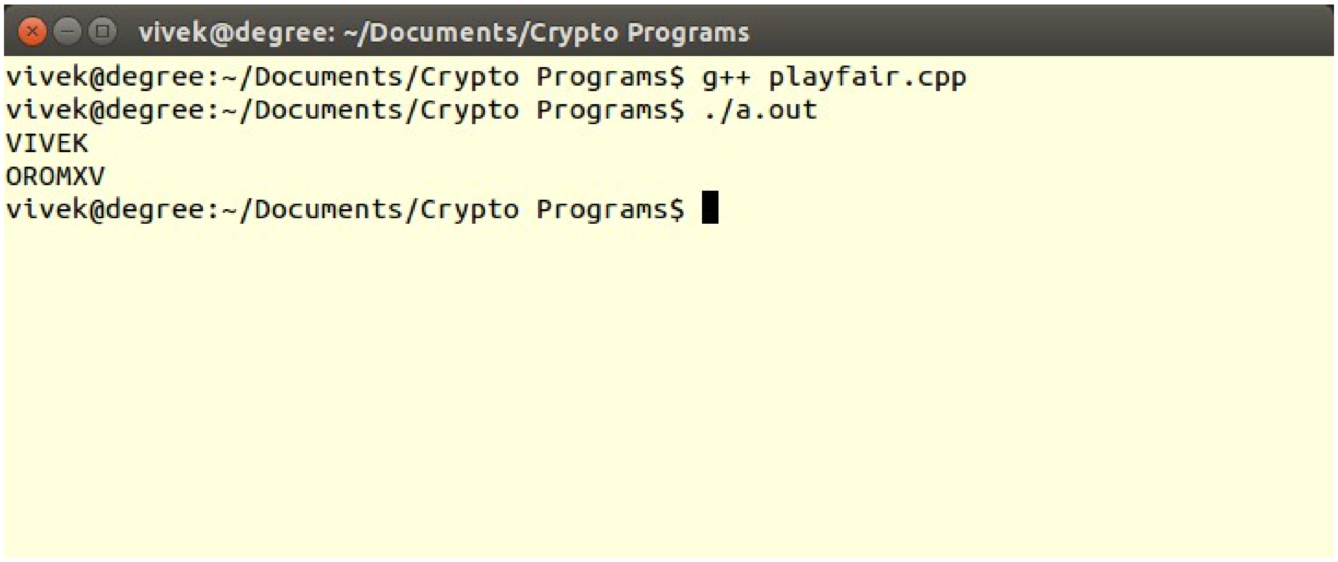


```
vivek@degree:~/lab$ g++ decrptionAffine.cpp
vivek@degree:~/lab$ ./a.out
VIVEK
Enter the decription key5 3
OBOVR
vivek@degree:~/lab$
```

6.Play Fair Cipher

```
#include<stdio.h>
#include<string.h>
char Table[5][5]={{'L','G','D','B','A'},{'Q','M','H','E','C'},{'U','R','N','I','F'},{'X','V','S','O','K'},
{'Z','Y','W','T','P'}};
void findTable(char p,int *a ,int *b)
{
    int i,j;
    for(i=0;i<5;i++) {
        for(j=0;j<5;j++) {
            if(Table[i][j]==p) {
                *(a)=i;
                *(b)=j;
                return;
            }
        }
    }
}
void findcord(char *p,char *q,char a,char b)
{
    int i,j,r,s;
    findTable(a,&i,&j);
    findTable(b,&r,&s);
    int x,y,k,l;
    if(i==r) {
        x=i; y=j+1;  y%=5; k=r; l=s+1; l%=5;
    }
    else if(j==s)
    {
        x=i+1; x%=5;    y=j;   k=r+1;
        k%=5;
        l=s;
    }
    else
    {
        x=i;
        y=s;
        k=r;
        l=j;
    }
    *(p)=Table[x][y];
    *(q)=Table[k][l];
}
int main()
{
    printf("Enter the plain text!n");
    char plaintext[101];
    char ciphertext[102];
    scanf("%s",plaintext);
    int n=strlen(plaintext);
    char p,q;
    char a,b;
    int i;
    for(i=0;i<n;i++)
    {
        a=plaintext[i++];
        if(i==n)
        {
            b='A';
        }
        else
        b=plaintext[i];
        findcord(&p,&q,a,b);
        ciphertext[i-1]=p;
        ciphertext[i]=q;
    }
    ciphertext[i]='';
    printf("Cipher text is %sn",ciphertext);
}
```

input-output:



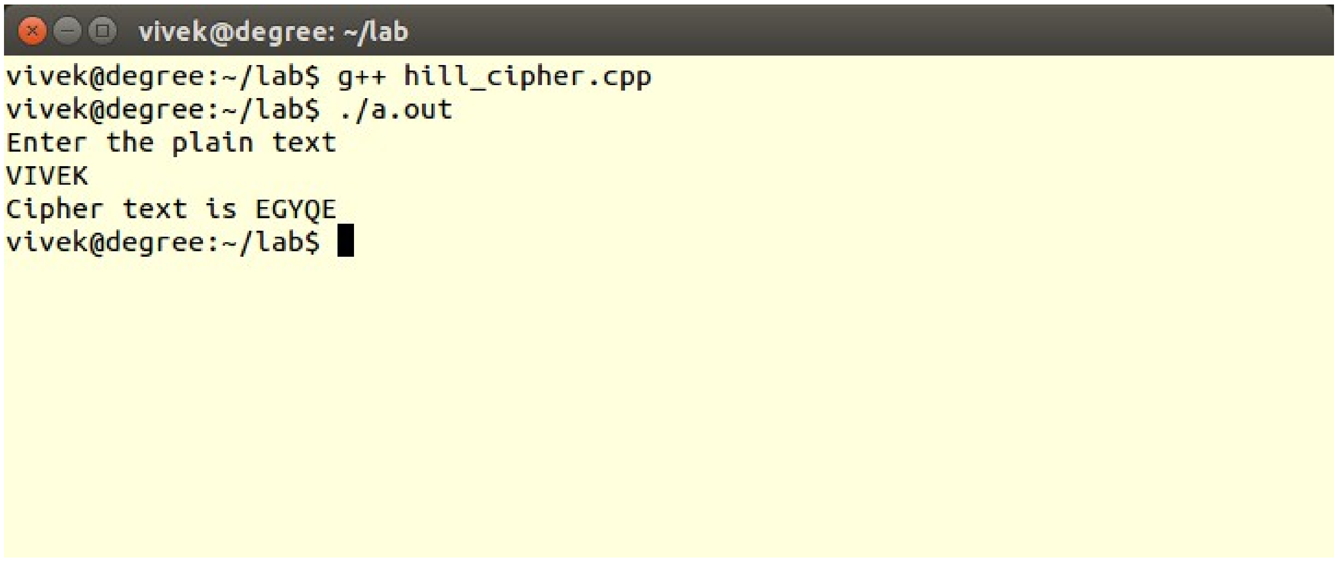
vivek@degree: ~/Documents/Crypto Programs

```
vivek@degree:~/Documents/Crypto Programs$ g++ playfair.cpp
vivek@degree:~/Documents/Crypto Programs$ ./a.out
VIVEK
OROMXV
vivek@degree:~/Documents/Crypto Programs$ 
```

7.Hill Cipher

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
int keyTable[3][3]={{3,5,7},{9,11,15},{17,19,21}};
int main()
{
    int j;
    printf("Enter the plain text!n");
    char plaintext[101];
    char ciphertext[102];
    scanf("%s",plaintext);
    int n=strlen(plaintext);
    int *sol[3];
    int *result[3];
    int m=n/3+(n%3!=0);
    char p,q;
    char a,b;
    int i;
    int temp;
    for(i=0;i<3;i++)
    sol[i]=(int*)malloc(m*sizeof(int));
    for(i=0;i<3;i++)
    result[i]=(int*)malloc(m*sizeof(int));
    int k=0;
    for(i=0;i<3;i++)
    {
        for(j=0;j<m;j++)
        {
            if(k<n)
            {
                sol[i][j]=plaintext[k]-'A';
                k++;
            }
            else
            {
                sol[i][j]=2;
            }
        }
    }
    for(i=0;i<3;i++)
    {
        for(j=0;j<m;j++)
        {
            int ans=0;
            for(k=0;k<3;k++)
            {
                ans+=keyTable[i][k]*sol[k][j];
            }
            ans%=26;
            result[i][j]=ans;
        }
    }
    k=0;
    for(i=0;i<3;i++)
    {
        for(j=0;j<m;j++)
        {
            ciphertext[k++]=result[i][j]+'A';
            if(k>=n)
            break;
        }
        if(k>=n)
        break;
    }
    ciphertext[k]='';
    printf("Cipher text is %sn",ciphertext);
}
```

input-output:

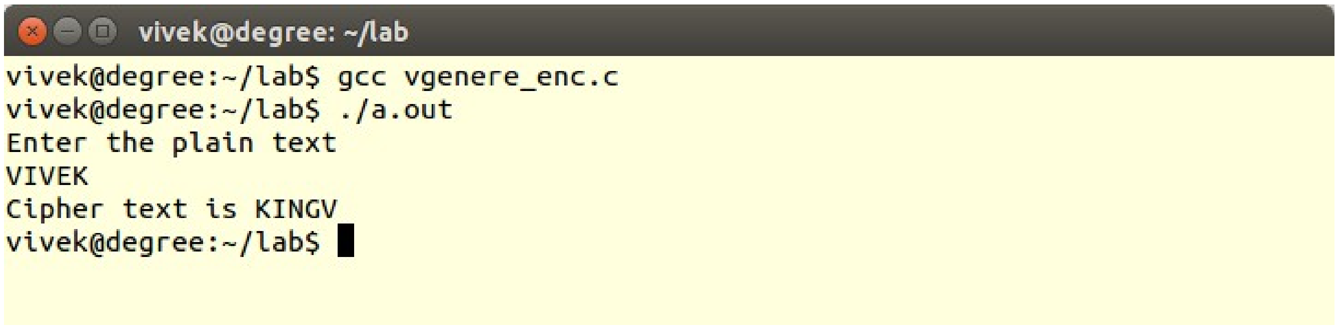


```
vivek@degree:~/lab$ g++ hill_cipher.cpp
vivek@degree:~/lab$ ./a.out
Enter the plain text
VIVEK
Cipher text is EGYQE
vivek@degree:~/lab$
```

8.Vigenere Cipher

```
#include<stdio.h>
#include<string.h>
int key[5]={15,0,18,2,11};
int main()
{
   int j;
   printf("Enter the plain text!n");
   char plaintext[101];
   char ciphertext[102];
   scanf("%s",plaintext);
   int n=strlen(plaintext);
   char p,q;
   char a,b;
   int i;
   int temp;
   for(i=0;i<n;)
   {
      for(j=0;j<5;j++)
      {
         temp=plaintext[i]-'A'+key[j];
         temp%=26;
         printf("%dn",j);
         ciphertext[i]=temp+'A';
         i++;
         if(i>=n) break;
      }
   }
   ciphertext[i]='';
   printf("Cipher text is %sn",ciphertext);
}
```

input-output:



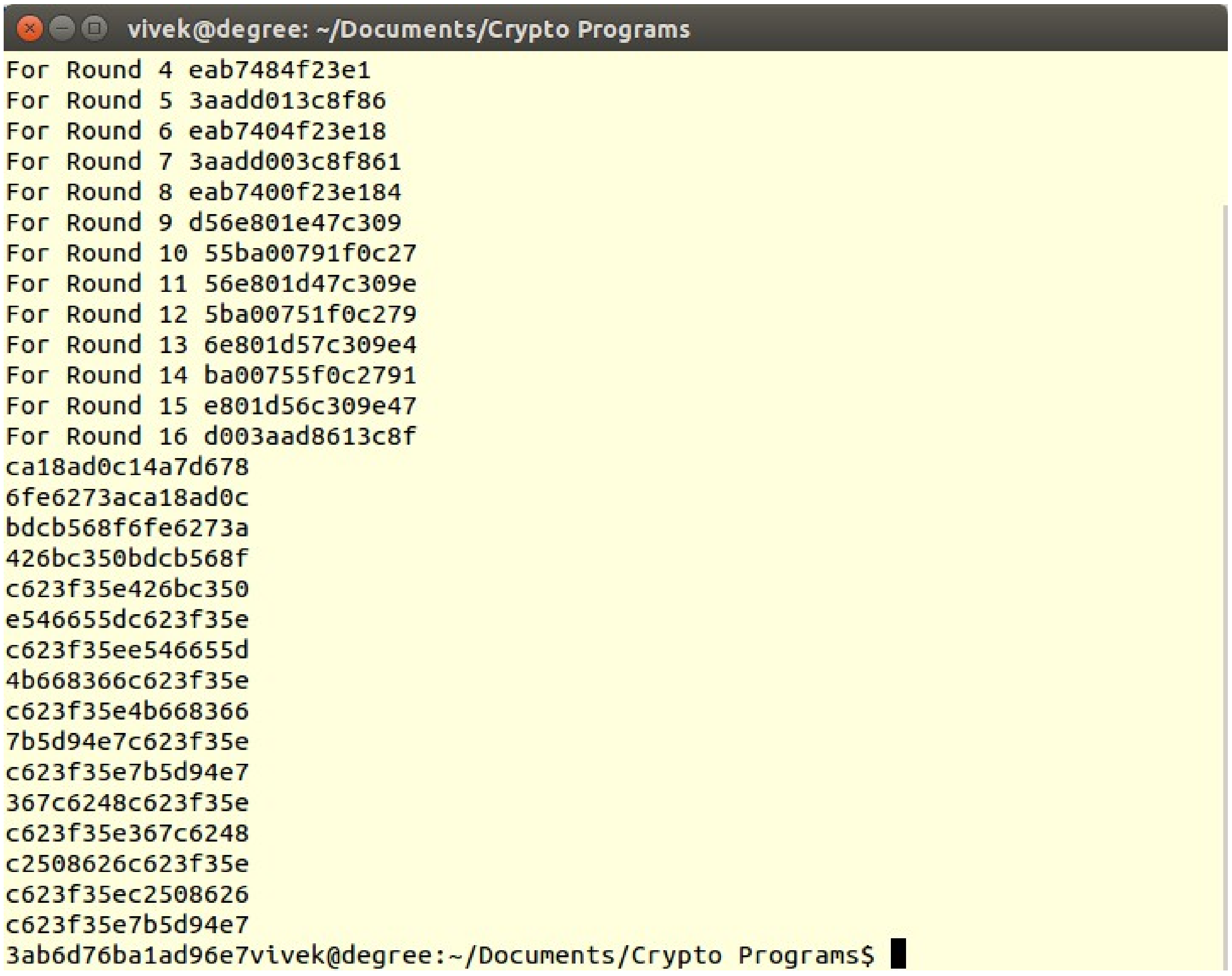
```
vivek@degree: ~/lab
vivek@degree:~/lab$ gcc vgenere_enc.c
vivek@degree:~/lab$ ./a.out
Enter the plain text
VIVEK
Cipher text is KINGV
vivek@degree:~/lab$
```

9.DES

```
#include<bits/stdc++.h>
using namespace std;
int IP[] = {58, 50, 42, 34,26, 18, 10, 2, 60, 52, 44, 36, 28, 20, 12, 4, 62, 54, 46, 38, 30, 22, 14, 6,
                 64, 56, 48, 40, 32, 24, 16, 8, 57, 49, 41, 33, 25, 17, 9, 1, 59, 51, 43, 35, 27, 19, 11, 3,
                61, 53, 45, 37, 29, 21, 13, 5, 63, 55, 47, 39, 31, 23, 15, 7};
int FP[] = {40, 8, 48, 16, 56, 24, 64, 32,  39, 7, 47, 15, 55, 23, 63, 31,  38, 6, 46, 14, 54, 22, 62, 30,
                37, 5, 45, 13, 53, 21, 61, 29, 36, 4, 44, 12, 52, 20, 60, 28,  35, 3, 43, 11, 51, 19, 59, 27,
                34, 2, 42, 10, 50, 18, 58, 26, 33, 1, 41, 9, 49, 17, 57, 25};
int expension[] = { 32, 1, 2, 3, 4, 5,  4, 5, 6, 7, 8, 9, 8, 9, 10, 11, 12, 13, 16, 17, 18, 19, 20, 21,
                              24, 25, 26, 27, 28, 29, 28, 29, 31, 31, 32, 1};
int parity[] = {57, 49, 41, 33, 25, 17, 9, 1, 58, 50, 42, 34, 26, 18,  10, 2, 59, 51, 43, 35, 27,
                      19, 11, 3, 60, 52, 44, 36,  63, 55, 47, 39, 31, 23, 15, 7, 62, 54, 46, 38, 30, 22,
                      14, 6, 61, 53, 45, 37, 29, 21, 13, 5, 28, 20, 12, 4};
int cmpress[] = {14, 17, 11, 24, 1, 5, 3, 28, 15, 6, 21, 10, 23, 19, 12, 4, 26, 8, 16, 7, 27, 20, 13, 2,
                          41, 52, 31, 37, 47, 55, 30, 40, 51, 45, 33, 48, 44, 49, 39, 56, 34, 53,
                          46, 42, 50, 36, 29, 32};
int strtperm[] = {16, 7, 20, 21, 29, 12, 28, 17,  1, 15, 23, 26, 5, 18, 31, 10, 2, 8, 24, 14, 32, 27, 3, 9,
                           19, 13, 30, 6, 22, 11, 4, 25 };
int s1[][16] = {{14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5, 9, 0, 7},
                         {0, 15, 7, 4, 14, 2, 13, 1, 10, 6, 12, 11, 9, 5, 3, 8},
                         {4, 1, 14, 8, 13, 6, 2, 11, 15, 12, 9, 7, 3, 10, 5, 0},
                         {15, 12, 8, 2, 4, 9, 1, 7, 5, 11, 3, 14, 10, 0, 6, 13}};
int s2[][16] = {{15, 1, 8, 14, 6, 11, 3, 4, 9, 7, 2, 13, 12, 0, 5, 10},
                        {3, 13, 4, 7, 15, 2, 8, 14, 12, 0, 1, 10, 6, 9, 11, 5},
                        {0, 14, 7, 11, 10, 4, 13, 1, 5, 8, 12, 6, 9, 3, 2, 15},
                       {13, 8, 10, 1, 3, 15, 4, 2, 11, 6, 7, 12, 0, 5, 14, 9}};
int s3[][16] = {{10, 0, 9, 14, 6, 3, 15, 5, 1, 13, 12, 7, 11, 4, 2, 8},
                        {13, 7, 0, 9, 3, 4, 6, 10, 2, 8, 5, 14, 12, 11, 15, 1},
                        {13, 6, 4, 9, 8, 15, 3, 0, 11, 1, 2, 12, 5, 10, 14, 7},
                        {1, 10, 13, 0, 6, 9, 8, 7, 4, 15, 14, 3, 11, 5, 2, 12} };
int s4[][16] = {{7, 13, 14, 3, 0, 6, 9, 10, 1, 2, 8, 5, 11, 12, 4, 15},
                       {13, 8, 11, 5, 6, 15, 0, 3, 4, 7, 2, 12, 1, 10, 14, 9},
                       {10, 6, 9, 0, 12, 11, 7, 13, 15, 1, 3, 14, 5, 2, 8, 4},
                       {3, 15, 0, 6, 10, 1, 13, 8, 9, 4, 5, 11, 12, 7, 2, 14} };
int s5[][16] = {{2, 12, 4, 1, 7, 10, 11, 6, 8, 5, 3, 15, 13, 0, 14, 9},
                       {14, 11, 2, 12, 4, 7, 13, 1, 5, 0, 15, 10, 3, 9, 8, 6},
                       {4, 2, 1, 11, 10, 13, 7, 8, 15, 9, 12, 5, 6, 3, 0, 14},
                       {11, 8, 12, 7, 1, 14, 2, 13, 6, 15, 0, 9, 10, 4, 5, 3} };
int s6[][16] = {{12, 1, 10, 15, 9, 2, 6, 8, 0, 13, 3, 4, 14, 7, 5, 11},
                        {10, 15, 4, 2, 7, 12, 9, 5, 6, 1, 13, 14, 0, 11, 3, 8},
                        {9, 14, 15, 5, 2, 8, 12, 3, 7, 0, 4, 10, 1, 13, 11, 6},
                        {4, 3, 2, 12, 9, 5, 15, 10, 11, 14, 1, 7, 6, 0, 8, 13}  };
int s7[][16] = {{4, 11, 2, 14, 15, 0, 8, 13, 3, 12, 9, 7, 5, 10, 6, 1},
                       {13, 0, 11, 7, 4, 9, 1, 10, 14, 3, 5, 12, 2, 15, 8, 6},
                        {1, 4, 11, 13, 12, 3, 7, 14, 10, 15, 6, 8, 0, 5, 9, 2},
                         {6, 11, 13, 8, 1, 4, 10, 7, 9, 5, 0, 15, 14, 2, 3, 12}       };
int s8[][16] = {{13, 2, 8, 4, 6, 15, 11, 1, 10, 9, 3, 14, 5, 0, 12, 7},
                         {1, 15, 13, 8, 10, 3, 7, 4, 12, 5, 6, 11, 0, 14, 9, 2},
                        {7, 11, 4, 1, 9, 12, 14, 2, 0, 6, 10, 13, 15, 3, 5, 8},
                        { 2, 1, 14, 7, 4, 10, 8, 13, 15, 12, 9, 0, 3, 5, 6, 11}};

void getIp(char *binr, char *ipr)
{
        int i;
        for(i=0; i<64; i++)     ipr[i] = binr[IP[i]-1];
}
void getParity(char *ipr, char *party)
{
        int i;
        for(i=0; i<56; i++)    party[i] = ipr[parity[i]-1];
}
void getLeft(char *party, char *left)
{
        int i;
        for(i=0; i<28; i++)    left[i] = party[i];
}
void getRight(char *party, char *right)
{
        int i;
        for(i=28; i<56; i++)    right[i-28] = party[i];
}
void leftShift(char *arr)
{
        int i;
        char ch = arr[0];
        for(i=0; i<27; i++)   arr[i] = arr[i+1];
       arr[i] = ch;
}
void hexToBin(string str, char binr[])
{
        int i = 0,len = str.size();
        while(i<len) {
                int t;
                if(str[i]>='0' && str[i]<='9')     t = str[i]-'0';
                if(str[i]=='a' || str[i] == 'A')       t = 10;
                if(str[i]=='b' || str[i] == 'B')       t = 11;
                if(str[i] == 'c' || str[i] == 'C')     t = 12;
                if(str[i]=='d' || str[i] == 'D')       t = 13;
                if(str[i] == 'e' || str[i] == 'E')      t =14;
                if(str[i] == 'f' || str[i] == 'F')       t = 15;
                int k = 3;
                while(k>=0)
                {
                        char ch = '0'+t%2;
                        binr[i*4+k] = ch;  k--;  t = t/2;
                }
                i++;
        } }
void expensionptr(char str[], char expnr[])
{
        int i;
        for(i=0; i<48; i++)   expnr[i] = str[expension[i]-1];
}
void xorr(char str[], char ptr[])
{
        int i;
        for(i=0; i<48; i++)
                if(str[i]==ptr[i])
                        str[i] = '0';
                else
                        str[i] = '1';
}
void placeBit(char str[], int num, int t)
{
        int k =3;
        while(k>=0) {
                char ch = '0'+num%2;  str[4*t+k] = ch; num = num/2;
                k--;
        }
}
 void getRC(char expnr[], int i, int &r, int &c)
{
        r = (expnr[i]-'0')*2+(expnr[i+5]-'0');
        c = (expnr[i+1]-'0')*8+(expnr[i+2]-'0')*4+(expnr[i+3]-'0')*2+(expnr[i+4]-'0');
}
void getStrS(char expnr[], char str[])
{
        int i, j, r, c;
        getRC(expnr, 0, r, c);
        int num, t = 0;
        num = s1[r][c];
        placeBit(str, num, t);   getRC(expnr, 6, r, c); num = s2[r][c]; t++;
        placeBit(str, num, t);  t++;  getRC(expnr, 12, r, c);  num = s3[r][c];
        placeBit(str, num, t);  t++;  getRC(expnr, 18, r, c);   num = s4[r][c];
        placeBit(str, num, t); t++; getRC(expnr, 24, r, c);  num = s5[r][c];
        placeBit(str, num, t);   t++;  getRC(expnr, 30, r, c);  num = s6[r][c];
        placeBit(str, num, t); t++; getRC(expnr, 36, r, c); num = s5[r][c];
        placeBit(str, num, t); t++; getRC(expnr, 42, r, c); num = s6[r][c];
        placeBit(str, num, t);   t++;

}
void straightD(char str[], char right[])
{
        int i;
        for(i=0; i<32; i++)  right[i] = str[strtperm[i]-1];
}
```

```
void getFP(char str[], char fp[])
{
        int i;
        for(i=0; i<64; i++)  {
                fp[i] = str[FP[i]-1]; cout<<fp[i];
        }
}
int main()
{
        char binr[64];
        string str = "123456abcd132536",key = "AABB09182736CCDD";
        int len = str.size(), i = 0, k;
        hexToBin(key, binr);
        char ipr[64], party[56];
        getIp(binr, ipr);  getParity(ipr, party);
        char left[30], right[30];
        getLeft(party, left); getRight(party, right);
        char keyr[17][49]; i=1;
        while(i<=16)  {
                if(i==1 || i==2 || i==9 || i==16) {
                        leftShift(left);  leftShift(right);
                }
                else   {
                        leftShift(left); leftShift(left);
                        leftShift(right);leftShift(right);
                }
                for(k=0; k<48; k++) {
                        if(cmpress[k]>27)
                                keyr[i][k] = right[cmpress[k]-28];
                        else
                                keyr[i][k] = left[cmpress[k]];
                }
                i++;
        }
        char bstr[65];
        hexToBin(str, bstr);
        char leftp[33], rightp[33];
        for(i=0; i<32; i++)    leftp[i] = bstr[i];
        leftp[32] = '\0';
        for(i=0; i<32; i++)     ightp[i] = bstr[i+32];
        rightp[64] = '\0';
        i = 1;
       while(i<=16) {
                char expnr[49], strs[33];
                expensionptr(rightp, expnr);
                xorr(expnr, keyr[i]); getStrS(expnr, strs); straightD(strs, rightp);
                char temp[33]; strcpy(temp, rightp);  temp[32] = '\0';
                strcpy(rightp, leftp);   rightp[32] = '\0'; strcpy(leftp, temp);     leftp[32] = '\0';
                i++;  }
        char finp[66]; leftp[32] = '\0'; rightp[64] = '\0';
        strcpy(finp, left);  strcat(finp, rightp);
        cout<<finp<<endl;
        char fp[65]; cout<<"Final Result: ";
        getFP(finp, fp);  cout<<endl;
        return 0;
}
```

<u>input-output:</u>



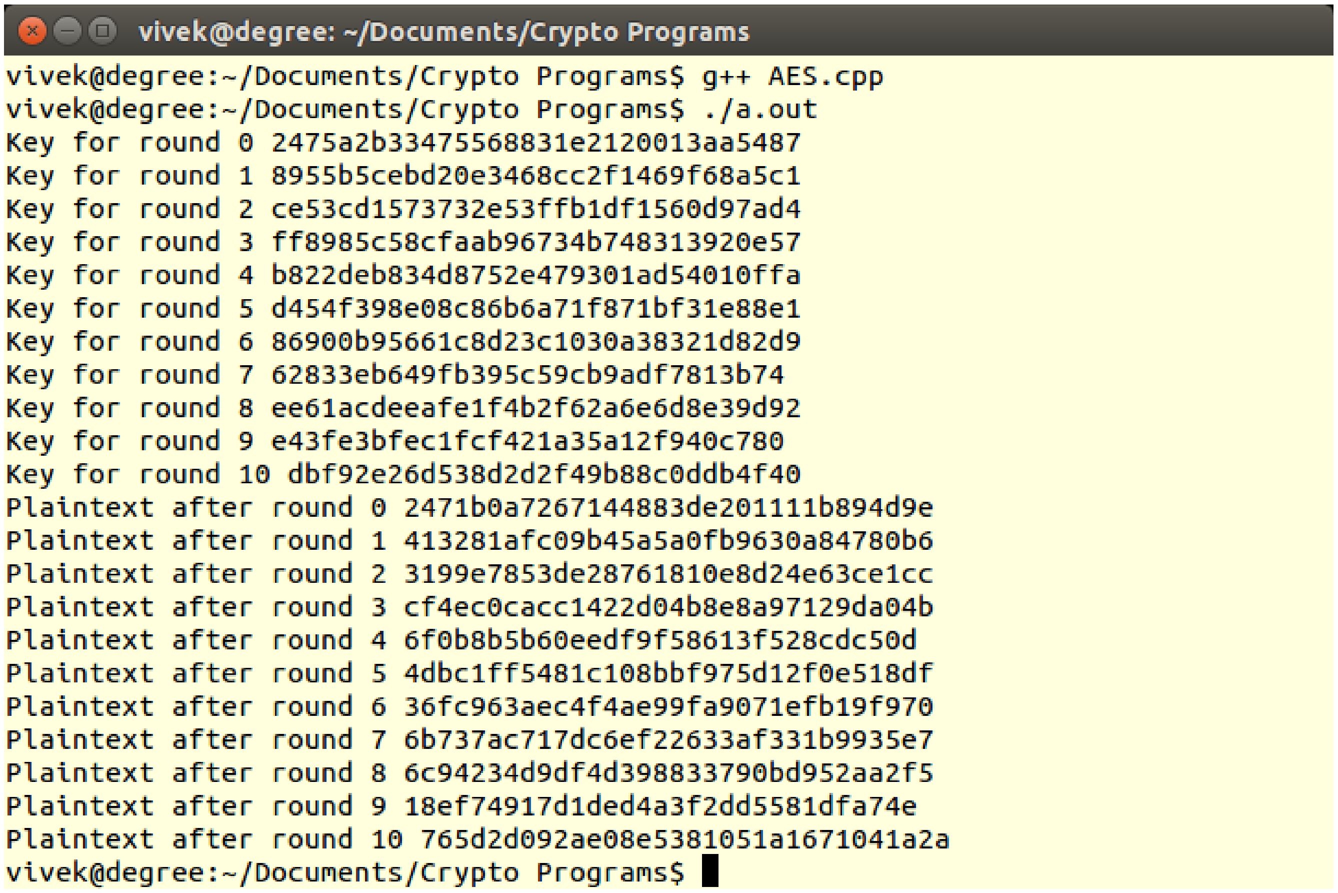
```
For Round 4 eab7484f23e1
For Round 5 3aadd013c8f86
For Round 6 eab7404f23e18
For Round 7 3aadd003c8f861
For Round 8 eab7400f23e184
For Round 9 d56e801e47c309
For Round 10 55ba00791f0c27
For Round 11 56e801d47c309e
For Round 12 5ba00751f0c279
For Round 13 6e801d57c309e4
For Round 14 ba00755f0c2791
For Round 15 e801d56c309e47
For Round 16 d003aad8613c8f
ca18ad0c14a7d678
6fe6273aca18ad0c
bdcb568f6fe6273a
426bc350bdcb568f
c623f35e426bc350
e546655dc623f35e
c623f35ee546655d
4b668366c623f35e
c623f35e4b668366
7b5d94e7c623f35e
c623f35e7b5d94e7
367c6248c623f35e
c623f35e367c6248
c2508626c623f35e
c623f35ec2508626
c623f35e7b5d94e7
3ab6d76ba1ad96e7vivek@degree:~/Documents/Crypto Programs$
```

10.AES

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
int sbox[256] ={ 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};
char RC[10][8] = {{'0','0','0','0','0','0','0','1'}, {'0','0','0','0','0','0','1','0'}, {'0','0','0','0','0','1','0','0'},
                  {'0','0','0','0','1','0','0','0'}, {'0','0','0','1','0','0','0','0'}, {'0','0','1','0','0','0','0','0'},
                  {'0','1','0','0','0','0','0','0'}, {'1','0','0','0','0','0','0','0'}, {'0','0','0','1','1','0','1','1'},
                  {'0','0','1','1','0','1','1','0'} };
char getHexDigit(char a,char b,char c,char d)
{
   char ar[4]; ar[0]=a; ar[1]=b; ar[2]=c; ar[3]=d;
   if(strcmp(ar,"0000")==0)   return '0';
   else if(strcmp(ar,"0001")==0)  return '1';
   else if(strcmp(ar,"0010")==0)  return '2';
   else if(strcmp(ar,"0011")==0)  return '3';
   else if(strcmp(ar,"0100")==0)  return '4';
   else if(strcmp(ar,"0101")==0)  return '5';
   else if(strcmp(ar,"0110")==0) return '6';
   else if(strcmp(ar,"0111")==0) return '7';
   else if(strcmp(ar,"1000")==0) return '8';
   else if(strcmp(ar,"1001")==0) return '9';
   else if(strcmp(ar,"1010")==0) return 'A';
   else if(strcmp(ar,"1011")==0)  return 'B';
   else if(strcmp(ar,"1100")==0)  return 'C';
   else if(strcmp(ar,"1101")==0) return 'D';
   else if(strcmp(ar,"1110")==0) return 'E';
   else  return 'F';
}
char xor(char x,char y)
{
   if(x==y) return '0';
   else  return '1';
}
```

```
void xorArr(char a[],char b[],char c[],int n)
{
   int i=0;
   for(i=0;i<n;i++)   c[i]=xor(a[i],b[i]);
}
char ** keygeneration(char str[])
{
   char **word;
   word=(char**)malloc(4*sizeof(char*));
   int i,j;
   for(i=0;i<4;i++)  word[i]=(char*)malloc(32*sizeof(char));
   for(i=0;i<4;i++)
      for(j=0;j<32;j++)
         word[i][j]=str[i*32+j];
   return word;
}
void intToString(int n,char V[])
{
      int i;
      for(i=7;i>=0;i--)  \V[i]=(char)(n%2+'0');  n/=2;
 }
void S(char V1[])
{
      int index=stringToInt(V1,8),x=sbox[index];
      intToString(x,V1);
}
void delta(char A[],char B[],int round)
{
   char V[4][8];
   int i,j;
   for(i=0;i<4;i++) for(j=0;j<8;j++)  V[(i-1+4)%4][j]=A[i*8+j];
    S(V[0]); S(V[1]);  S(V[2]); S(V[3]);
    xorArr(V[0],RC[round],V[0],8);
   for(i=0;i<4;i++) for(j=0;j<8;j++)  B[8*i+j]=V[i][j];
}
\void schedule(char **word,char **temp,int round)
{
   char deltaVal[32];
   delta(word[3],deltaVal,round);
   xorArr(word[0],deltaVal,temp[0],32); xorArr(temp[0],word[1] ,temp[1],32);
   xorArr(temp[1],word[2] ,temp[2],32); xorArr(temp[2],word[3] ,temp[3],32);
}

int main()
{
   char input[128];
   scanf("%s",input);
   char **word,**temp;
   word=keygeneration(input);
   int i,j,k;
   for( i=0;i<4;i++) \printf("%s\n",word[i]);
   temp=(char**)malloc(4*sizeof(char*));
  for(i=0;i<4;i++)  temp[i]=(char*)malloc(32*sizeof(char));
   for(k=0;k<10;k++) {
   printf("Round %d\n",k+1);
   schedule(word,temp,k);
   for(i=0;i<4;i++) {
                for(j=0;j<8;j++)
                  printf("%c",getHexDigit(temp[i][j*4+0],temp[i][j*4+1],temp[i][j*4+2],temp[i][j*4+3]));
              strcpy(word[i],temp[i]);
         }  }
   return 0;
}
```

input-output:



```
vivek@degree:~/Documents/Crypto Programs$ g++ AES.cpp
vivek@degree:~/Documents/Crypto Programs$ ./a.out
Key for round 0 2475a2b33475568831e2120013aa5487
Key for round 1 8955b5cebd20e3468cc2f1469f68a5c1
Key for round 2 ce53cd1573732e53ffb1df1560d97ad4
Key for round 3 ff8985c58cfaab96734b748313920e57
Key for round 4 b822deb834d8752e479301ad54010ffa
Key for round 5 d454f398e08c86b6a71f871bf31e88e1
Key for round 6 86900b95661c8d23c1030a38321d82d9
Key for round 7 62833eb649fb395c59cb9adf7813b74
Key for round 8 ee61acdeeafe1f4b2f62a6e6d8e39d92
Key for round 9 e43fe3bfec1fcf421a35a12f940c780
Key for round 10 dbf92e26d538d2d2f49b88c0ddb4f40
Plaintext after round 0 2471b0a7267144883de201111b894d9e
Plaintext after round 1 413281afc09b45a5a0fb9630a84780b6
Plaintext after round 2 3199e7853de28761810e8d24e63ce1cc
Plaintext after round 3 cf4ec0cacc1422d04b8e8a97129da04b
Plaintext after round 4 6f0b8b5b60eedf9f58613f528cdc50d
Plaintext after round 5 4dbc1ff5481c108bbf975d12f0e518df
Plaintext after round 6 36fc963aec4f4ae99fa9071efb19f970
Plaintext after round 7 6b737ac717dc6ef22633af331b9935e7
Plaintext after round 8 6c94234d9df4d398833790bd952aa2f5
Plaintext after round 9 18ef74917d1ded4a3f2dd5581dfa74e
Plaintext after round 10 765d2d092ae08e5381051a1671041a2a
vivek@degree:~/Documents/Crypto Programs$ 
```

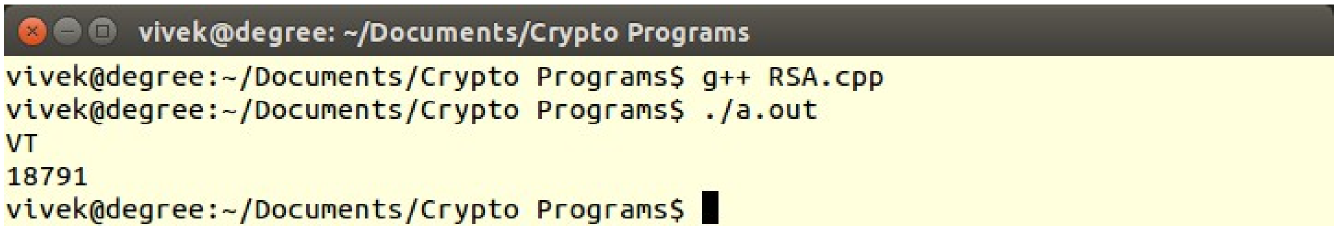
-

11.RSA

```
#include<bits/stdc++.h>
using namespace std;
int power1(int a,int b,int mod)
{
        long long int x=a,y=1;
        while(b) {
        if(b&1)  y=(y*x)%mod;
        x=(x*x)%mod;  b>>=1;
        }
        return (int)y;
}
int modInverse(int a, int m) {
         a %= m;
         for(int x = 1; x < m; x++) if((a*x) % m == 1) return x;
}
void keyGenerationAlgorithm(int &e,int &d,int &n)
{
        int p,q,phi_n;
        cin>>p>>q;
        n=p*q; phi_n=(p-1)*(q-1);
         cin>>e;
         d=modInverse(e,phi_n);
          return;
}
int RsaEncryption(int p,int e,int n)
{
        return power1(p,e,n);
}
string inttostring(int number)
{
        string A="";
        while(number) {
                 int temp=number%100; A=(char)(temp+'a')+A; number/=100;
                }
         return A;
}
int RsaDecryption(int c,int d,int n)
{
        return (power1(c,d,n));
}
int stringtoint(string A)
{
        int n=A.size(),int number=0;
        for(int i=0;i<n;i++) {
                int temp=(A[i]-'a');  number=number*100+temp;

         }
 return number; }
```

```
int main()
{
            int e,d,n;
            e=0,d=0,n=1;
            keyGenerationAlgorithm(e,d,n);
           string plaintext;
            cin>>plaintext;
            int C=RsaEncryption(stringtoint(plaintext),e,n);
            int t=RsaDecryption(C,d,n);
            cout<<inttostring(t)<<endl;

}
```

<u>input-output:</u>



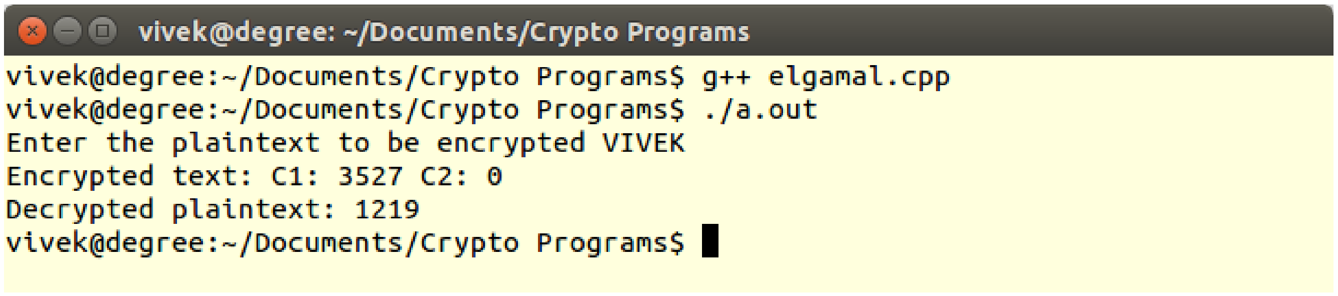
```
vivek@degree:~/Documents/Crypto Programs$ g++ RSA.cpp
vivek@degree:~/Documents/Crypto Programs$ ./a.out
VT
18791
vivek@degree:~/Documents/Crypto Programs$ 
```

12.Elgamal Cipher

```
#include<bits/stdc++.h>
using namespace std;
int modPower(int a, int p, int n)
{
	int ans = 1;
	while(p) {
		if(p&1)  ans = (ans*a)%n;
		a = (a*a)%n;  p = p/2;
	}
	return ans;
}
int primitive(int n)
{
	for(int a=2;a<n; a++) {
		int count = 0;
		for(int i=1; i<n; i++) {
			if(modPower(a, i, n)==1)  count++;
		}
		if(count==1) return a;
	}
}
int main()
{
	int p,d;
	cin>>p;
	int e1 = primitive(p);
	srand(time(NULL));
	while(1)  {
		d = rand()%p;
		if(d>0 && d<p-1) break;
	}
	int e2 = modPower(e1, d, p);
	int r = rand()%p;
	int c1 = modPower(e1, r, p);
	int msg;
	cin>>msg;
	int z = modPower(e2, r, p);
	int c2 = (msg*z)%p;
	cout<<"Encrypttion: ";
	cout<<c1<<" "<<c2<<endl;
	cout<<"Decryption: ";
	int cr = modPower(c1, d, p);
	int cz = 1;
	while(1)
	{
		if((cz*cr-1)%p==0)
			break;
		cz++;
```

```
        }
        int msz = c2*cz%p;
        cout<<msz<<endl;
        return 0;
}
```

input-output:

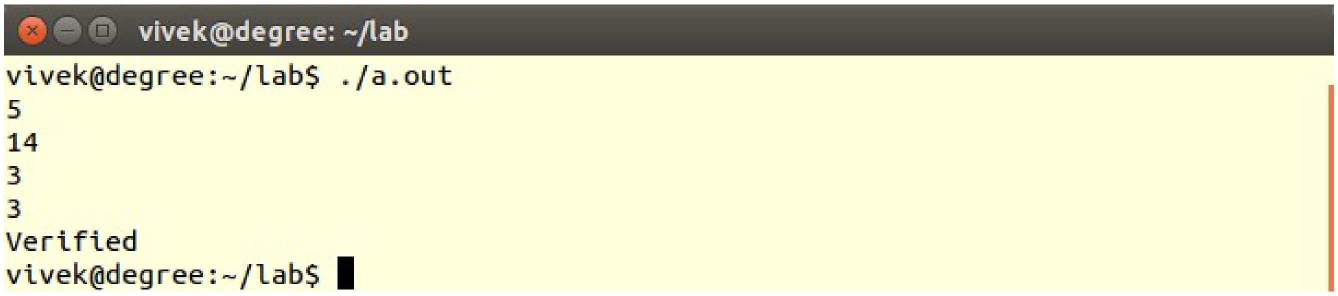


```
vivek@degree:~/Documents/Crypto Programs$ g++ elgamal.cpp
vivek@degree:~/Documents/Crypto Programs$ ./a.out
Enter the plaintext to be encrypted VIVEK
Encrypted text: C1: 3527 C2: 0
Decrypted plaintext: 1219
vivek@degree:~/Documents/Crypto Programs$
```

13.Elegmal Digital Signature:

```
#include<bits/stdc++.h>
using namespace std;

int modPower(int a, int p, int n)
{
        int ans = 1;
        while(p) {
                if(p&1) ans = (ans*a)%n;
                a = (a*a)%n;   p = p/2;
        }
        return ans;
}

int primitive(int n)
{
        for(int a=2;a<n; a++)  {
                int count = 0;
                for(int i=1; i<n; i++)  {
                   if(modPower(a, i, n)==1)  count++;
                }
                if(count==1)  return a;
        }
}
bool isVerify(int e1, int e2, int s1, int s2, int msg, int p)
{
        int res = modPower(e2, s1, p);
        int v2 = (res*modPower(s1, s2, p))%p;
        int v1 = modPower(e1, msg, p);
        if(v1==v2) return true;
        return false;
}

int main()
{
        int p;
        cin>>p;
        int e1 = primitive(p);
        int msg;
        cin>>msg;
        srand(time(NULL));
        int r = 307; //rand()%p;
        int d = 127;
        while(1) {
                d = rand()%p;
                if(d>0 && d<p-1)   break;
        }

        int e2 = modPower(e1, d, p);
        int s1 = modPower(e1, r, p);
        int s2 = (msg-d*s1);
        int inv = 1;
        while(1)   {
                if((inv*r-1)%(p-1)==0)   break;
                inv++;
        }
        s2 = (s2*inv)%(p-1);
        if(s2<0) s2+=(p-1);
        if(isVerify(e1, e2, s1, s2, msg, p))
                cout<<"Verified"<<endl;
        else
                cout<<"Not Verified"<<endl;
        return 0;
}
```

<u>input-output:</u>



```
vivek@degree: ~/lab
vivek@degree:~/lab$ ./a.out
5
14
3
3
Verified
vivek@degree:~/lab$ 
```

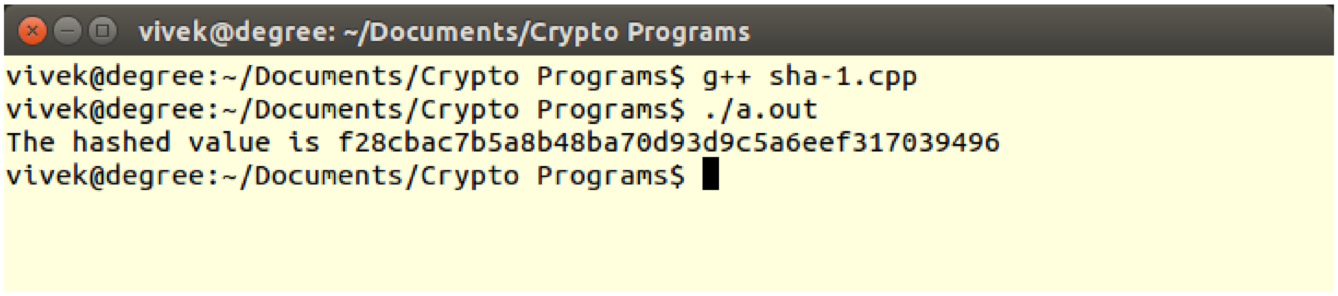
14. SHA

```
#include <bits/stdc++.h>
using namespace std;
typedef unsigned int uint;
vector<uint> plaintext;
vector<uint> words;
uint H0 = 0x67452301, H1 = 0xefcdab89, H2 = 0x98badcfe, H3 = 10325476, H4 = 0xc3d2e1f0,k
uint leftShift(uint x, int n) {
        if(n > 32)  n %= 32;
        uint y = (x << n) | (x >> (32 - n));
        return y;
}
void createWords() {
        int i;
        for(i = 0; i < plaintext.size(); i++)
                        words.push_back(plaintext[i]);
        for(i = 16; i < 80; i++) {
                uint wordT = leftShift(words[i - 3] ^ words[i - 8] ^ words[i - 14] ^ words[i - 16], 1);
                words.push_back(wordT);
        }
}
uint functionT(uint B, uint C, uint D, int t) {
        if(t < 20) {
                k = 0x5a827999;
                return (B & C) | (~(B) & D);
        } else if(t < 40) {
                k = 0x6ed9eba1;
                return B ^ C ^ D;
        } else if(t < 60) {
                k = 0x8f1bbcdc;
                return (B & C) | (B & D) | (C & D);
        } else {
                k = 0xca6261d6;
                return B ^ C ^ D;
        }
}
void performHash() {
 uint A = H0, B = H1, C = H2, D = H3, E = H4;
        int i;
        for(i = 0; i < 80; i++) {
                uint t = leftShift(A, 5) + functionT(B, C, D, i) + E + words[i] + k;
                E = D; D = C;C = leftShift(B, 30);
                B = A;A = t;
        }
        H0 = H0 + A; H1 = H1 + B;
        H2 = H2 + C;H3 = H3 + D;
        H4 = H4 + E;
}
```

```
uint convertToHex(char ch) {
	if(ch >= '0' && ch <= '9')
		return 0x0 + ch - '0';
	else
		return 0xa + ch - 'a';
}
int main() {
	string str = "aaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbcccccccccc";
	int i , length = 0;
	uint t;
	for(i = 0; i < str.size(); i += 8) {
		t = convertToHex(str[i]);
		length += 4;
		int flag = 0;
		if(i + 1 < str.size()) {
			t <<= 4;
			t |= convertToHex(str[i + 1]);
			length += 4;
		}
		if(i + 2 < str.size()) {
			t <<= 4;
			t |= convertToHex(str[i + 2]);
			length += 4;
		}
		if(i + 3 < str.size()) {
			t <<= 4;
			t |= convertToHex(str[i + 3]);
			length += 4;
		}
		if(i + 1 < str.size()) {
			t <<= 4;
			t |= convertToHex(str[i + 1]);
			length += 4;
		}
		if(i + 2 < str.size()) {
			t <<= 4;
			t |= convertToHex(str[i + 2]);
			length += 4;
		}
		if(i + 3 < str.size()) {
			t <<= 4;
			t |= convertToHex(str[i + 3]);
			length += 4;
		}
		if(flag)
			plaintext.push_back(t);
	}
	int size = length;
	if(length < 448) {
```

```
            if(length % 32 == 8) {
                  t <<= 24;
                  t |= (1 << 24);
                  length += 24;
                  plaintext.push_back(t);
            } else if(length % 32 == 16) {
                  t <<= 16;
                  t |= (1 << 16);
                  length += 16;
                  plaintext.push_back(t);
            } else if(length % 32 == 24) {
                  t <<= 8;
                  t |= (1 << 8);
                  length += 8;
                  plaintext.push_back(t);
            }
      }
      for(; length < 448; length += 4) {
            plaintext.push_back(0);
      }
      plaintext.push_back(0);
      plaintext.push_back(size);
      for(i = 0; i < plaintext.size(); i++) {
            cout << hex << plaintext[i] << endl;
      }
      createWords();
      performHash();
      cout << "The hashed value is ";
      cout << hex << H0 << H1 << H2 << H3 << H4 << endl;
      return 0;
}
```

<u>input-output:</u>

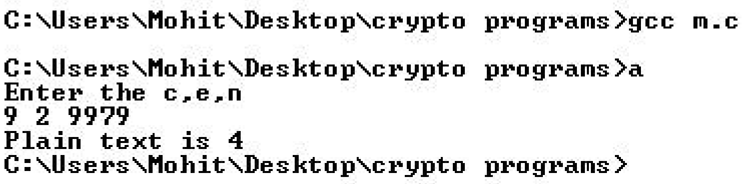


```
vivek@degree:~/Documents/Crypto Programs$ g++ sha-1.cpp
vivek@degree:~/Documents/Crypto Programs$ ./a.out
The hashed value is f28cbac7b5a8b48ba70d93d9c5a6eef317039496
vivek@degree:~/Documents/Crypto Programs$ 
```

15.Chosen Ciphertext Attack on RSA

```
#include <bits/stdc++.h>
using namespace std;
bool isPrime(int a) {
        for (int i = 2; i * i <= a; ++i)  if (a % i == 0) return false;
        return true;
}
vector<int> getMeTuple(int a) {
        vector<int> primes (2, 0);
        for (int i = 2; i <= a; ++i)
            if (a % i == 0 && isPrime(i) && isPrime(a / i)) {
                        primes[0] = i; primes[1] = a / i;
                }
       return primes;
}
int fastExpo(int a, int b, int MOD) {
        if (b == 0) return 1;
        int result = fastExpo(a, b >> 1, MOD);  result = (result * result) % MOD;
        if (b & 1) result = (result * (a % MOD)) % MOD;
        return result;
}
int inverse(int e, int MOD) {
        int result = 0;
        while (true) {
                if ((e * result) % MOD == 1) return result;
                ++result;
        }
        return result;
}
int main() {
        int p, e, n, c;
        cin >> p >> c >> e >> n;
        vector<int> myPQ = getMeTuple(n);
       int d = inverse(e, (myPQ[0] - 1) * (myPQ[1] - 1));
        int decrypted = fastExpo(c, d, n);
        cout << (decrypted == p) << endl;
        cout << decrypted << endl;
        return 0;
}
```

input-output:

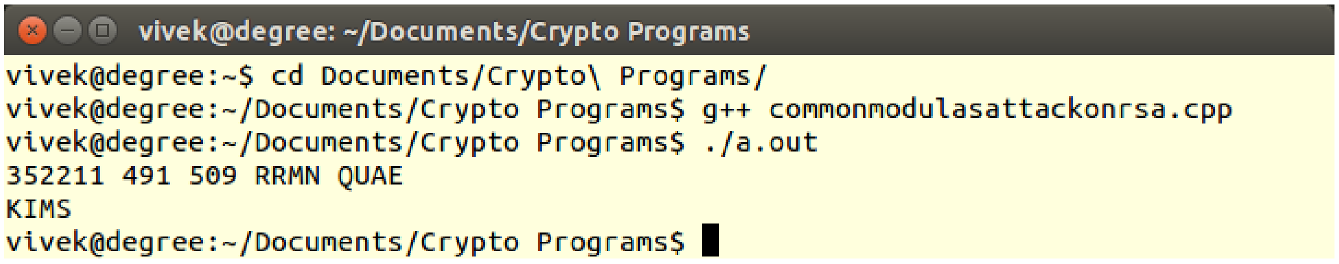


```
C:\Users\Mohit\Desktop\crypto programs>gcc m.c

C:\Users\Mohit\Desktop\crypto programs>a
Enter the c,e,n
9 2 9979
Plain text is 4
C:\Users\Mohit\Desktop\crypto programs>
```

<u>16.Common Modulas Attack on RSA</u>

```
#include <iostream>
using namespace std;
long long int e, f, n,cffs, cffb;
void calcInverse(long long int a, long long int b) {
        long long int newa = b, newb = a % b;
        if(newa % newb == 0) {
                cffs = (a / b) * (-1);
                cffb = 1;
               return;
        }
        calcInverse(b, a % b);
        long long int store = cffs;
        cffs = cffb - (a / b) * (cffs);
        cffb = store;
}
long long int power(long long int plaintext, long long int e) {
            long long int ans = 1;
            while(e) {
                if(e & 1)
                        ans = (ans * plaintext) % n;
                plaintext = (plaintext * plaintext) % n;
                e = e >> 1;
         }
        return ans;
}
long long int convertToNum(string str) {
        int i;
        long long int ans = 0;
        for(i = 0; i < str.size(); i++)   ans = (ans * 26 + str[i] - 65) % n;
     return ans;
}
int main() {
        cin >> n,e,f;
        string cipher1, cipher2;
        string ans = "";
        long long cipherNum1, cipherNum2, plaintext;
        cin >> cipher1>>cipher2;
        cipherNum1 = convertToNum(cipher1);
        cipherNum2 = convertToNum(cipher2);
        //cout << cipherNum1 << " " << cipherNum2 << endl;
        long long int x, y;
        if(e < f) {
                calcInverse(f, e);
                y = cffb; x = cffs;
                x = x % n; y = y % n;
                if(x < 0)   x += f;
                if(y < 0)   y += e;
         else {
                calcInverse(e, f);
                y = cffs;x = cffb;

        }
        if(y < 0) {
                calcInverse(n, cipherNum2);
                plaintext = (power(cipherNum1, x) * power(cffs, -y)) % n;
        } else if(x < 0) {
                calcInverse(n, cipherNum1);
                plaintext = (power(cffs, -x) * power(cipherNum2, y)) % n;
        }
       while(plaintext) {
                int rem = plaintext % 26;string s = "";
                s.push_back(rem + 65);
                ans = s + ans; plaintext /= 26;
        }
        cout << ans << endl;
        return 0;
}
```

input-output:



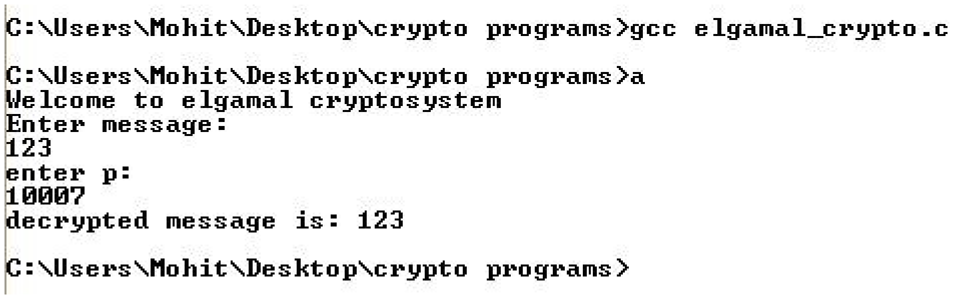
```
vivek@degree:~$ cd Documents/Crypto\ Programs/
vivek@degree:~/Documents/Crypto Programs$ g++ commonmodulasattackonrsa.cpp
vivek@degree:~/Documents/Crypto Programs$ ./a.out
352211 491 509 RRMN QUAE
KIMS
vivek@degree:~/Documents/Crypto Programs$
```

17.Elliptical Cryptosystem

```
#include <bits/stdc++.h>
using namespace std;
typedef pair<int,int> point;
int a = 2, b = 3;
int p = 67;

int inverse(int v) {
        //cout<<"INv "<<v<<endl;
        if(v < 0)
                v += p;
        int i = 1;
        while(i<p) {
                if( (v*i) %p ==1 )
                        return i;
                i++;
        }
}

point multiply(point e1, int d) {
        point p1 = e1;
        point p2 = e1;

        for(int i=0;i<d-1;i++) {
                int x1 = p1.first;
                int y1 = p1.second;
                int x2 = p2.first;
                int y2 = p2.second;

                if(x1 == x2 && y1 == -1*y2)
                        cout<<"here\n";
                else if(x1 == x2 && y1 == y2) {
                        int lamda =( (3*x1*x1 + a)*inverse( (2*y1)%p ) )%p ;
                        p2.first =  ((lamda*lamda - x1 - x2 )%p )%p ;
                        p2.second = ((lamda*(x1 - p2.first) - y1)%p  )%p;
                } else {
                        int t = x2 - y1;
                        if(t < 0)
                                t+=p;
                        int t1 = y2 - y1;
                        if(t1 < 0)
                                t1 += p;
                        int lamda = ( (t1)*inverse(t) )%p;
                        p2.first  = (( lamda*lamda - x1 - x2 )%p  )%p ;
                        p2.second = (( lamda*(x1-p2.first) -y1 )%p )%p;
                }

                cout<<i<<" "<<x1<<" "<<y1<<" "<<p2.first<<" "<<p2.second<<endl;
        }
```

```
	return p2;
}

int main() {

	point e1 = make_pair(2,22);
	int d = 4;
	point e2 = multiply(e1,d);
	cout<<e2.first<<" "<<e2.second;

	return 0;
}
```

input-output:

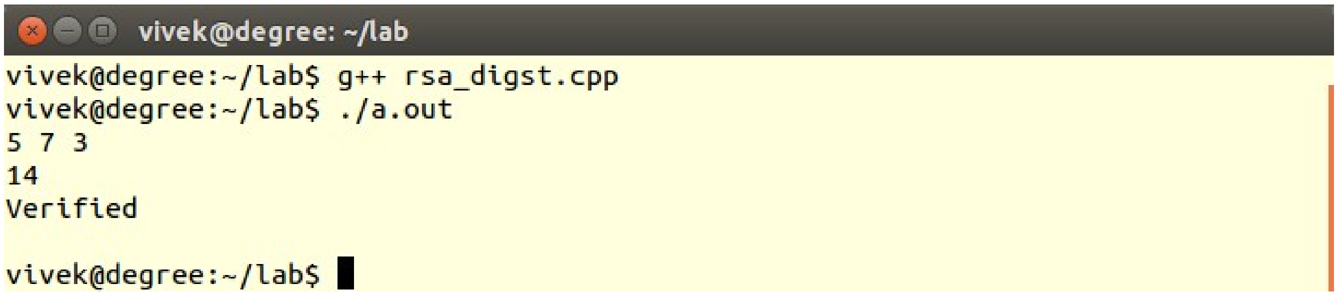


```
C:\Users\Mohit\Desktop\crypto programs>gcc elgamal_crypto.c

C:\Users\Mohit\Desktop\crypto programs>a
Welcome to elgamal cryptosystem
Enter message:
123
enter p:
10007
decrypted message is: 123

C:\Users\Mohit\Desktop\crypto programs>
```

18.RSA Digital Signature

```
#include<bits/stdc++.h>
using namespace std;
int modPower(int a, int p, int mod)
{
	long long ans = 1;
	while(p){
		if(p&1)  ans = (ans*a)%mod;
		a = (a*a)%mod;
		p = p/2;
	}
	return (int)ans;
}
int main()
{
	int p, q, e;  cin>>p>>q>>e;
	int msg; cin>>msg;
	int phi = (p-1)*(q-1);
	int n = p*q, d = 1;
	while(1){
		if(((d*e)-1)%phi==0)
			break;
		d++;
	}
	int y = modPower(msg, d, n), z = modPower(y, e, n);
	if(msg == z)
		cout<<"Verified";
	else
		cout<<"Not Verified";

	return 0;
}
```

input-output:



```
vivek@degree:~/lab$ g++ rsa_digst.cpp
vivek@degree:~/lab$ ./a.out
5 7 3
14
Verified

vivek@degree:~/lab$ 
```