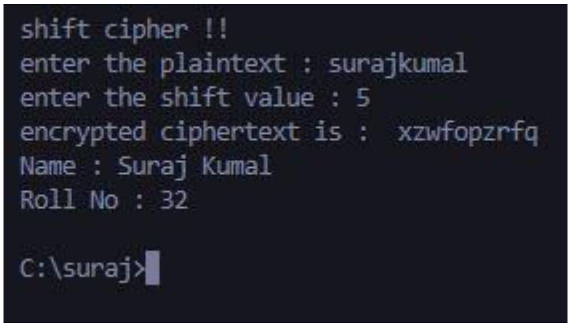
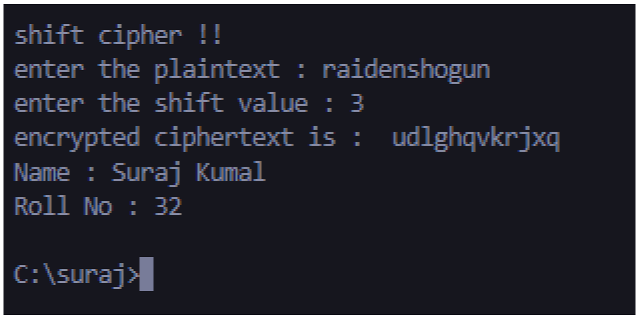
1. Program to implement Shift cipher.

```
#include<iostream>
#include<string>
using namespace std;
string encrypt(string text,int shift){
    string encryptedtxt = " ";
    for (char character : text) {
        if (isalpha(character)) {
            char shiftedCharacter = character + shift;
            if (isupper(character)) {
                if (shiftedCharacter > 'Z') {
                    shiftedCharacter -= 26;
                } else if (shiftedCharacter < 'A') {
                    shiftedCharacter += 26;
                }
            }
            else if (islower(character)) {
                if (shiftedCharacter > 'z') {
                    shiftedCharacter -= 26;
                } else if (shiftedCharacter < 'a') {
                    shiftedCharacter += 26;
                }
            }
            encryptedtxt += shiftedCharacter;
        } else {

            encryptedtxt += character;
        }
    }
    return encryptedtxt;
}
int main(){
    string text;
    int shift;
    cout<<"shift cipher !!"<<endl;
    cout<<"enter the plaintext : ";
    getline(cin,text);
    cout<<"enter the shift value : ";
    cin>>shift;
    string encryptedtext = encrypt(text,shift);
    cout<<"encrypted ciphertext is : "<<encryptedtext<<endl;
    cout<<"Name : Suraj Kumal"<<endl<<"Roll No : 32"<<endl;
    cin.get();
    return 0;
  }
```

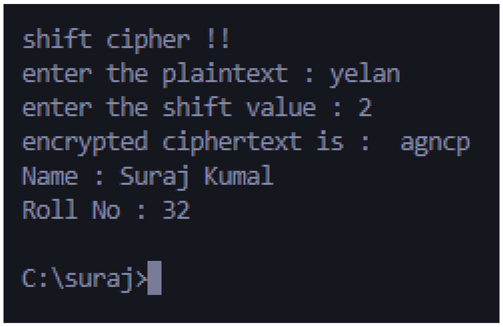
OUTPUT :



```
shift cipher !!
enter the plaintext : surajkumal
enter the shift value : 5
encrypted ciphertext is :  xzwfopzrfq
Name : Suraj Kumal
Roll No : 32

C:\suraj>
```



```
shift cipher !!
enter the plaintext : raidenshogun
enter the shift value : 3
encrypted ciphertext is :  udlghqvkrjxq
Name : Suraj Kumal
Roll No : 32

C:\suraj>
```



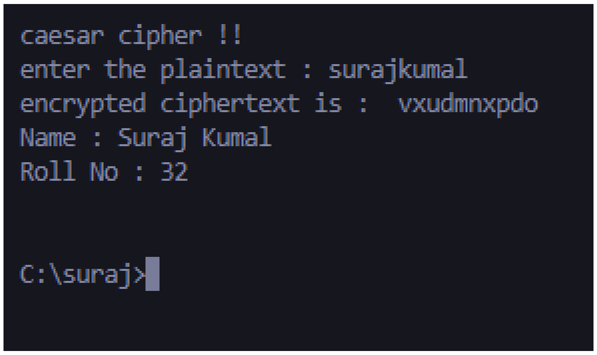
```
shift cipher !!
enter the plaintext : yelan
enter the shift value : 2
encrypted ciphertext is :  agncp
Name : Suraj Kumal
Roll No : 32

C:\suraj>
```

## 2. Program to implement Caesar cipher.

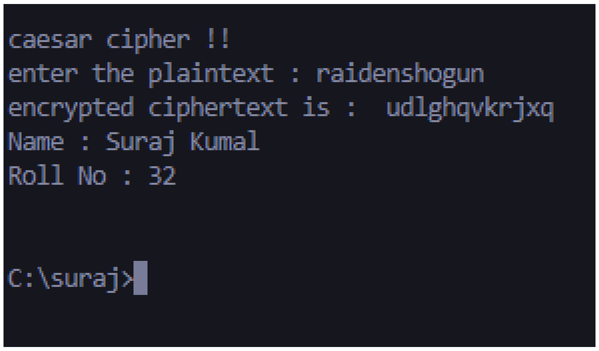
```
#include<iostream>
#include<conio.h>
#include<string>
using namespace std;
string encrypt(string text,int shift){
    string encryptedtxt = " ";
    for (char character : text) {
        if (isalpha(character)) {
            char shiftedCharacter = character + shift;
            if (isupper(character)) {
                if (shiftedCharacter > 'Z') {
                    shiftedCharacter -= 26;
                } else if (shiftedCharacter < 'A') {
                    shiftedCharacter += 26;
                }
            }
            else if (islower(character)) {
                if (shiftedCharacter > 'z') {
                    shiftedCharacter -= 26;
                } else if (shiftedCharacter < 'a') {
                    shiftedCharacter += 26;
                }
            }
            encryptedtxt += shiftedCharacter;
        } else {

            encryptedtxt += character;
        }
    }
    return encryptedtxt;
}
int main(){
    string text;
    int shift = 3;
    cout<<"caesar cipher !!"<<endl;
    cout<<"enter the plaintext : ";
    getline(cin,text);
    string encryptedtext = encrypt(text,shift);
    cout<<"encrypted ciphertext is : "<<encryptedtext<<endl;
    cout<<"Name : Suraj Kumal"<<endl<<"Roll No : 32"<<endl;
    cin.get();
    return 0;
```

}

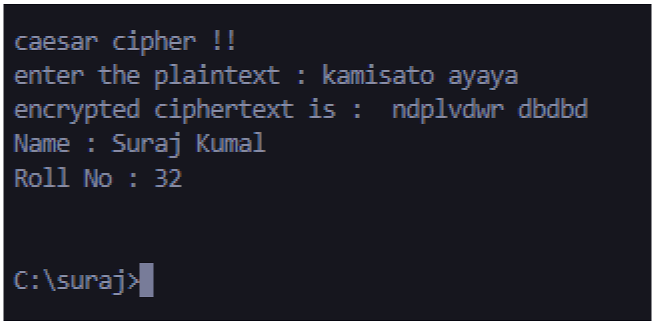
OUTPUT :



```
caesar cipher !!
enter the plaintext : surajkumal
encrypted ciphertext is :  vxudmnxpdo
Name : Suraj Kumal
Roll No : 32


C:\suraj>
```



```
caesar cipher !!
enter the plaintext : raidenshogun
encrypted ciphertext is :  udlghqvkrjxq
Name : Suraj Kumal
Roll No : 32


C:\suraj>
```

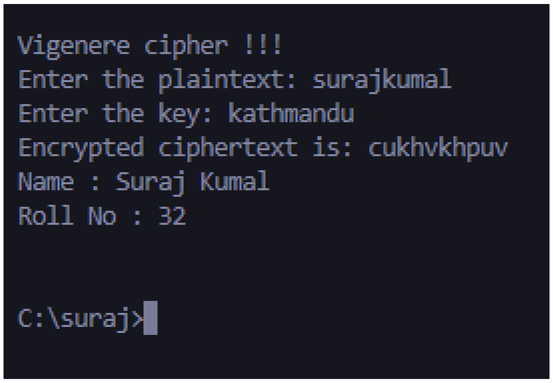


```
caesar cipher !!
enter the plaintext : kamisato ayaya
encrypted ciphertext is :  ndplvdwr dbdbd
Name : Suraj Kumal
Roll No : 32


C:\suraj>
```

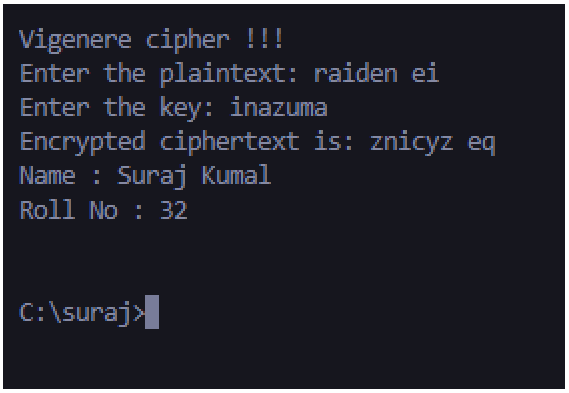
3. Program to implement Vigenere cipher.

```
#include <iostream>
#include <string>
using namespace std;
string encrypt(string text, string key) {
    string encryptedtxt = "";
    int keyIndex = 0;
    for (char character : text) {
        if (isalpha(character)) {
            char shift;
            if (isupper(character)) {
                shift = 'A';
            } else {
                shift = 'a';
            }
            char k_character = key[keyIndex % key.length()];
            char encryptedChar = ((character - shift + k_character - shift) % 26) + shift;
            encryptedtxt += encryptedChar;
            keyIndex++;
        } else {
            encryptedtxt += character;
        }
    }
    return encryptedtxt;
}

int main() {
    string text, key;
    cout << "Vigenere cipher !!!" << endl;
    cout << "Enter the plaintext: ";
    getline(cin, text);
    cout << "Enter the key: ";
    getline(cin, key);
    string encryptedtext = encrypt(text, key);
    cout << "Encrypted ciphertext is: " << encryptedtext << endl;
    cout<<"Name : Suraj Kumal"<<endl<<"Roll No : 32"<<endl;
    cin.get();
    return 0;
}
```

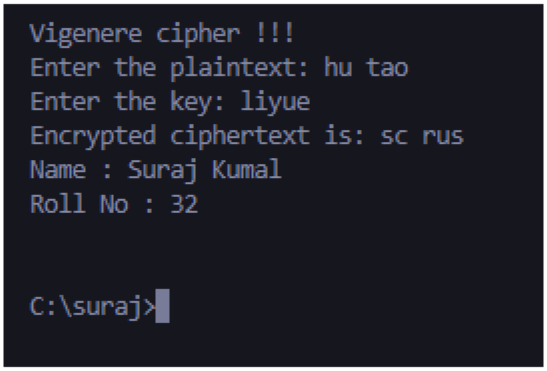
OUTPUT :



```
Vigenere cipher !!!
Enter the plaintext: surajkumal
Enter the key: kathmandu
Encrypted ciphertext is: cukhvkhpuv
Name : Suraj Kumal
Roll No : 32


C:\suraj>
```



```
Vigenere cipher !!!
Enter the plaintext: raiden ei
Enter the key: inazuma
Encrypted ciphertext is: znicyz eq
Name : Suraj Kumal
Roll No : 32


C:\suraj>
```



```
Vigenere cipher !!!
Enter the plaintext: hu tao
Enter the key: liyue
Encrypted ciphertext is: sc rus
Name : Suraj Kumal
Roll No : 32


C:\suraj>
```

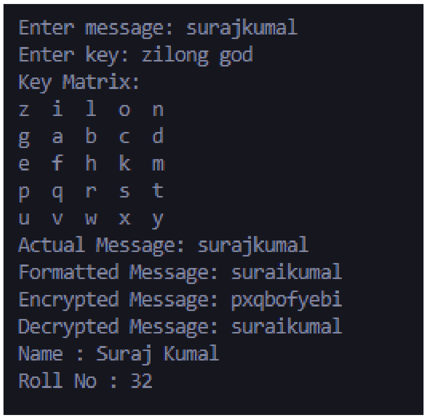
## 4. Program to implement Play fair cipher.

```
#include<iostream>
#include <bits/stdc++.h>
using namespace std;
typedef struct{
        int row;
        int col;
}position;

char mat[5][5]; // Global Variable

void generateMatrix(string key)
{
   /* flag keeps track of letters that are filled in matrix */
        /* flag = 0 -> letter not already present in matrix */
        /* flag = 1 -> letter already present in matrix */
   int flag[26] = {0};
   int x = 0, y = 0;

   /* Add all characters present in the key */
   for(int i=0; i<key.length(); i++)
   {
      if(key[i] == 'j') key[i] = 'i'; // replace j with i

      if(flag[key[i]-'a'] == 0)
      {
         mat[x][y++] = key[i];
         flag[key[i]-'a'] = 1;
      }
      if(y==5) x++, y=0;
   }

   /* Add remaining characters */
   for(char ch = 'a'; ch <= 'z'; ch++)
   {
      if(ch == 'j') continue; // don't fill j since j was replaced by i

      if(flag[ch - 'a'] == 0)
      {
         mat[x][y++] = ch;
         flag[ch - 'a'] = 1 ;
      }
      if(y==5) x++, y=0;
   }
}

/* function to add filler letter('x') */
string formatMessage(string msg)
{
   for(int i=0; i<msg.length(); i++)
   {
      if(msg[i] == 'j')  msg[i] = 'i';
   }

   for(int i=1; i<msg.length(); i+=2) //pairing two characters
   {
      if(msg[i-1] == msg[i])  msg.insert(i, "x");
   }

   if(msg.length()%2 != 0)  msg += "x";
   return msg;
}

/* Returns the position of the character */
position getPosition(char c)
{
   for(int i=0; i<5; i++)
   {
      for(int j=0; j<5; j++)
      {
         if(c == mat[i][j])
         {
            position p = {i, j};
            return p; // Return the position when a match is found
         }
      }
   }

   // If no matching character is found, return a default position
   position defaultPosition = {-1, -1};
   return defaultPosition;
}


string encrypt(string message)
{
   string ctext = "";
```

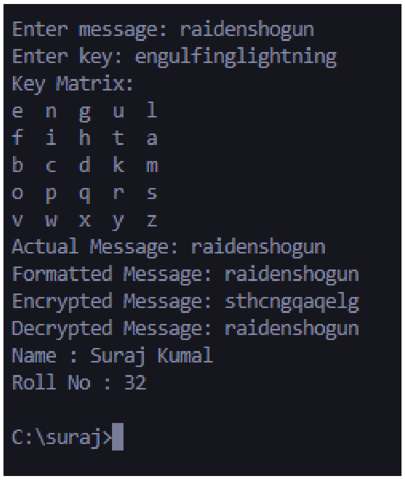
```
  for(int i=0; i<message.length(); i+=2)    // i is incremented by 2 inorder to check for
pair values
  {
              position p1 = getPosition(message[i]);
              position p2 = getPosition(message[i+1]);
    int x1 = p1.row; int y1 = p1.col;
    int x2 = p2.row; int y2 = p2.col;

    if( x1 == x2 ) // same row
    {
      ctext +=  mat[x1][(y1+1)%5];
      ctext +=  mat[x2][(y2+1)%5];
    }
    else if( y1 == y2 ) // same column
    {
      ctext += mat[ (x1+1)%5 ][ y1 ];
      ctext += mat[ (x2+1)%5 ][ y2 ];
    }
    else
    {
      ctext += mat[ x1 ][ y2 ];
      ctext += mat[ x2 ][ y1 ];
    }
  }
  return ctext;
}


string Decrypt(string message)
{
  string ptext = "";
  for(int i=0; i<message.length(); i+=2) // i is incremented by 2 inorder to check for
pair values
  {
    position p1 = getPosition(message[i]);
              position p2 = getPosition(message[i+1]);
    int x1 = p1.row; int y1 = p1.col;
    int x2 = p2.row; int y2 = p2.col;

    if( x1 == x2 ) // same row
    {
      ptext += mat[x1][ --y1<0 ? 4: y1 ];
      ptext += mat[x2][ --y2<0 ? 4: y2 ];
    }
    else if( y1 == y2 ) // same column
```

```
        {
            ptext += mat[ --x1<0 ? 4: x1 ][y1];
            ptext += mat[ --x2<0 ? 4: x2 ][y2];
        }
        else
        {
            ptext += mat[ x1 ][ y2 ];
            ptext += mat[ x2 ][ y1 ];
        }
    }
    return ptext;
}

int main()
{
    string plaintext;
    cout << "Enter message: ";
    cin >> plaintext;

    string key;
    cout << "Enter key: ";
    cin >> key;

    generateMatrix(key);

    cout << "Key Matrix:" << endl;
    for(int k=0; k<5; k++)
    {
        for(int j=0; j<5; j++)
        {
            cout << mat[k][j] << "  ";
        }
        cout << endl;
    }

    cout << "Actual Message: " << plaintext << endl;

    string fmsg = formatMessage(plaintext);
    cout << "Formatted Message: " << fmsg << endl;

    string ciphertext = encrypt(fmsg);
    cout << "Encrypted Message: " << ciphertext << endl;

    string decryptmsg = Decrypt(ciphertext);
    cout << "Decrypted Message: " << decryptmsg << endl;
    cout<<"Name : Suraj Kumal"<<endl<<"Roll No : 32"<<endl;
    return 0;
}
```

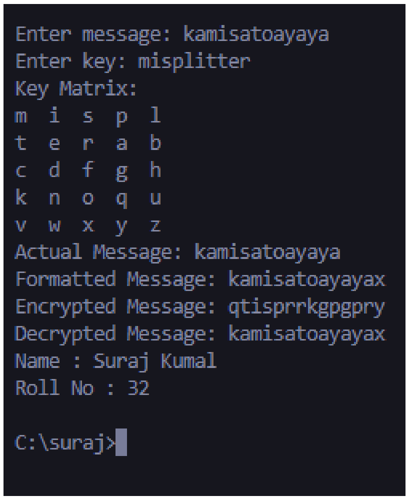
## OUTPUT :



```
Enter message: surajkumal
Enter key: zilong god
Key Matrix:
z  i  l  o  n
g  a  b  c  d
e  f  h  k  m
p  q  r  s  t
u  v  w  x  y
Actual Message: surajkumal
Formatted Message: suraikumal
Encrypted Message: pxqbofyebi
Decrypted Message: suraikumal
Name : Suraj Kumal
Roll No : 32
```



```
Enter message: raidenshogun
Enter key: engulfinglightning
Key Matrix:
e  n  g  u  l
f  i  h  t  a
b  c  d  k  m
o  p  q  r  s
v  w  x  y  z
Actual Message: raidenshogun
Formatted Message: raidenshogun
Encrypted Message: sthcngqaqelg
Decrypted Message: raidenshogun
Name : Suraj Kumal
Roll No : 32

C:\suraj>
```



```
Enter message: kamisatoayaya
Enter key: missplitter
Key Matrix:
m  i  s  p  l
t  e  r  a  b
c  d  f  g  h
k  n  o  q  u
v  w  x  y  z
Actual Message: kamisatoayaya
Formatted Message: kamisatoayayax
Encrypted Message: qtisprrkgpgpry
Decrypted Message: kamisatoayayax
Name : Suraj Kumal
Roll No : 32

C:\suraj>
```

## 5. Program to implement Rail fence cipher.

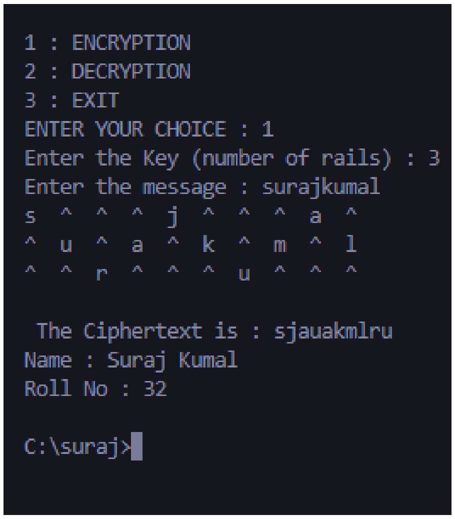
```
#include<iostream>
#include<string>
using namespace std;
class RailFence{
        public:
        int nrow,ncol;
        int getKey(){
                        int key;
                        cout<<"Enter the Key (number of rails) : ";
                        cin>>key;
                        return key;
                }
                string getMessage(){
                        string msg;
                        cout<<"Enter the message : ";
                        cin.ignore();
                        getline(cin,msg);
                        return msg;
                }
        void encrypt(string msg, int key){
                // creating a matrix to encrypt msg with key
          // key = rows , length of msg=no. of characters = columns
          nrow= key;
          ncol= msg.length();
          char rail_matrix[nrow][ncol];

          // filling the rail matrix with ^ symbol
          for (int i=0; i < nrow; i++) {
            for (int j = 0; j < ncol; j++){
              rail_matrix[i][j] ='^';
                        }
                }
                // to find the direction
          bool downward = false;
          int r = 0, c = 0;
          string ciphertext;

          for (int i=0; i < msg.length(); i++) {
            // checking  the direction of flow
            // reverse the direction if the top or bottom rail is just filled
            if (r == 0 || r == key-1)
              downward = !downward;
```

```
    // filling  with  characters in the plaintext
    rail_matrix[r][c++] = msg[i];

    // find the next row using direction
    downward ?r++ : r--;
  }
       //to print the rail matrix
       for (int i=0; i < nrow; i++) {
    for (int j = 0; j < ncol; j++){
     cout<< rail_matrix[i][j]<<"  ";
             }
             cout<<"\n";
       }
  // generating  the ciphertext using the rail_matrix

  for (int i=0; i < key; i++) {
    for (int j=0; j < msg.length(); j++) {
      if (rail_matrix[i][j]!='^')
        ciphertext.push_back(rail_matrix[i][j]); //appending a character
             }
       }

       cout<<"\n The Ciphertext is : "<<ciphertext<<"\n";
}

void decrypt(string msg, int key){

       // creating a matrix to encrypt msg with key
  // key = rows , length of msg=no. of characters = columns
  nrow= key;
  ncol= msg.length();
  char rail_matrix[nrow][ncol];
  string plaintext;

  // filling the rail matrix with ^ symbol
  for (int i=0; i < nrow; i++) {
    for (int j = 0; j < ncol; j++){
      rail_matrix[i][j] ='^';
             }
       }
       // to find the direction
  bool downward;
  int r = 0, c= 0;
```

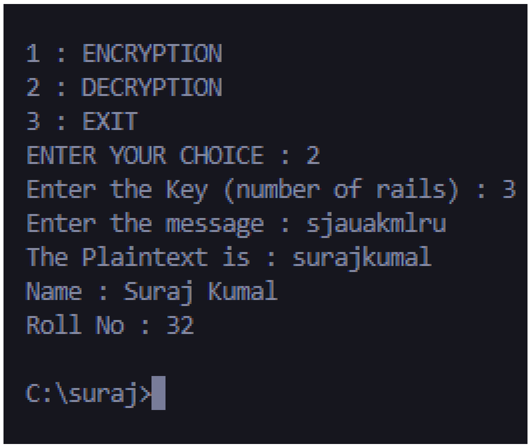
```
          // marking  the places with '~'
          for (int i=0; i < msg.length(); i++) {
            // check the direction of flow
            if (r == 0)
              downward = true;
            if (r == key-1)
              downward = false;
            // place the marker
            rail_matrix[r][c++] = '~';

            // find the next row using direction flag
            downward?r++ : r--;
          }

          // filling  the rail matrix
          int indx = 0;
          for (int i=0; i<key; i++) {
            for (int j=0; j<msg.length(); j++) {
              if (rail_matrix[i][j] == '~' && indx<msg.length())
                            rail_matrix[i][j] = msg[indx++];
               }
          }


          //  reading  the matrix in zig-zag order to get the plaintext
          r = 0, c = 0;
          for (int i=0; i< msg.length(); i++)
          {
            // check the direction of flow
            if (r == 0)
              downward = true;
            if (r == key-1)
              downward = false;

            // checking  the marker
            if (rail_matrix[r][c] != '~')
              plaintext.push_back(rail_matrix[r][c++]); //appending
            // finding  the next row using direction flag
            downward?r++: r--;
          }
        cout<<"The Plaintext is : "<<plaintext<<"\n";
        }


};
```

```
int main(){
    int choice;
    char more;
    RailFence rf;
    int k;
    string m;
        cout<<"\n1 : ENCRYPTION \n2 : DECRYPTION \n3 : EXIT \n";
  cout<<"ENTER YOUR CHOICE : ";
        cin>>choice;
        switch(choice){
            case 1:
                k= rf.getKey();
                m= rf.getMessage();
                rf.encrypt(m,k);
                break;

            case 2:
                k= rf.getKey();
                m= rf.getMessage();
                rf.decrypt(m,k);
                break;
            case 3:
                exit(1);
            default:
                cout<<"\n INVALID CHOICE! \n";
        }
 cout<<"Name : Suraj Kumal"<<endl<<"Roll No : 32"<<endl;
}
```

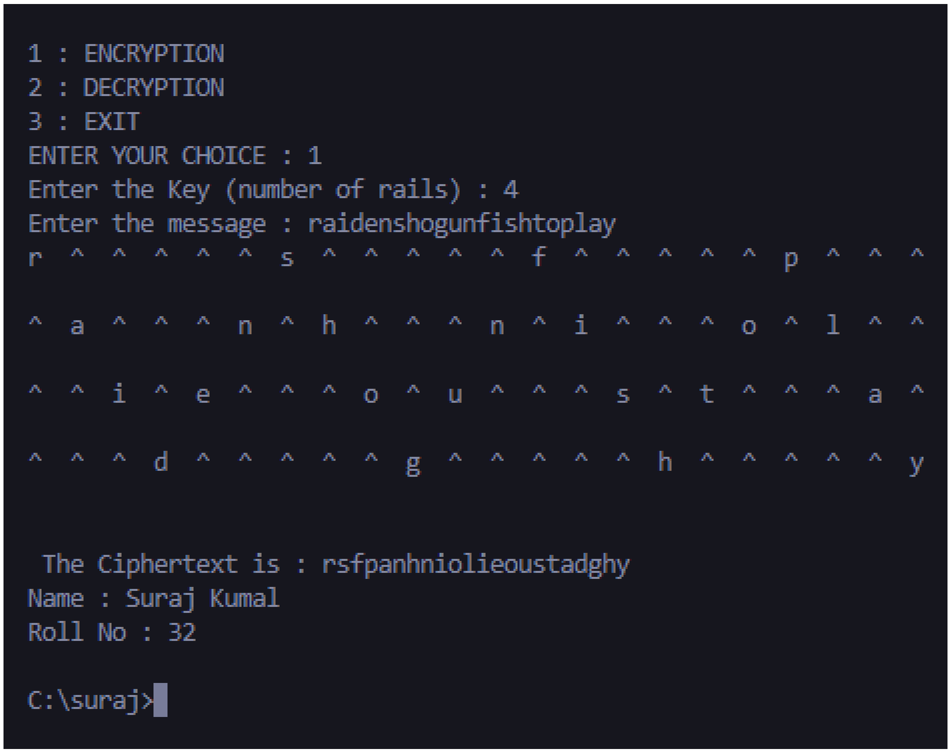
OUTPUT :



```
1 : ENCRYPTION
2 : DECRYPTION
3 : EXIT
ENTER YOUR CHOICE : 1
Enter the Key (number of rails) : 3
Enter the message : surajkumal
s  ^  ^  ^  j  ^  ^  ^  a  ^
^  u  ^  a  ^  k  ^  m  ^  l
^  ^  r  ^  ^  ^  u  ^  ^  ^

 The Ciphertext is : sjauakmlru
Name : Suraj Kumal
Roll No : 32

C:\suraj>
```



```
1 : ENCRYPTION
2 : DECRYPTION
3 : EXIT
ENTER YOUR CHOICE : 2
Enter the Key (number of rails) : 3
Enter the message : sjauakmlru
The Plaintext is : surajkumal
Name : Suraj Kumal
Roll No : 32

C:\suraj>
```

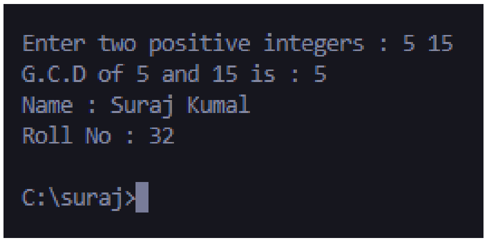


```
1 : ENCRYPTION
2 : DECRYPTION
3 : EXIT
ENTER YOUR CHOICE : 1
Enter the Key (number of rails) : 4
Enter the message : raidenshogunfishtoplay
r  ^  ^  ^  ^  ^  s  ^  ^  ^  ^  ^  f  ^  ^  ^  ^  ^  p  ^  ^  ^

^  a  ^  ^  ^  n  ^  h  ^  ^  ^  n  ^  i  ^  ^  ^  o  ^  l  ^  ^

^  ^  i  ^  e  ^  ^  ^  o  ^  u  ^  ^  ^  s  ^  t  ^  ^  ^  a  ^

^  ^  ^  d  ^  ^  ^  ^  ^  g  ^  ^  ^  ^  ^  h  ^  ^  ^  ^  ^  y


 The Ciphertext is : rsfpanhniolieoustadghy
Name : Suraj Kumal
Roll No : 32

C:\suraj>
```

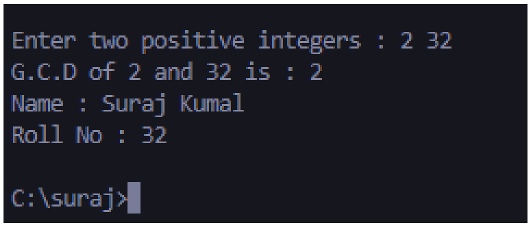
## 6. Program to compute GCD of two integers .

```
#include <iostream>
using namespace std;

int gcd(int n1, int n2)
{
   if(n2 != 0)
      return gcd(n2, n1 % n2);
   else
      return n1;
}
int main()
{
   int n1, n2;

   cout <<"Enter two positive integers : ";
   cin>> n1 >> n2;

   cout << "G.C.D of " << n1 << " and " << n2 << " is : " <<gcd(n1, n2)<<endl;
   cout<<"Name : Suraj Kumal"<<endl<<"Roll No : 32"<<endl;
   return 0;
}
```

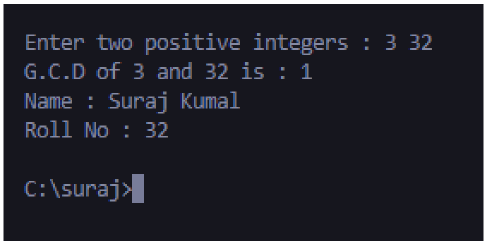
OUTPUT :



```
Enter two positive integers : 5 15
G.C.D of 5 and 15 is : 5
Name : Suraj Kumal
Roll No : 32

C:\suraj>
```



```
Enter two positive integers : 2 32
G.C.D of 2 and 32 is : 2
Name : Suraj Kumal
Roll No : 32

C:\suraj>
```

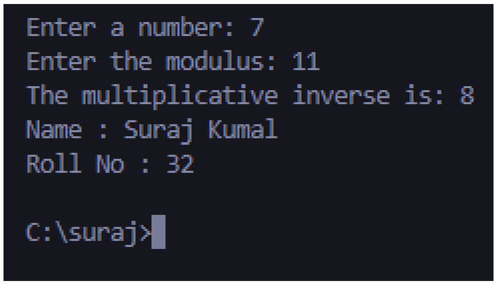


```
Enter two positive integers : 3 32
G.C.D of 3 and 32 is : 1
Name : Suraj Kumal
Roll No : 32

C:\suraj>
```

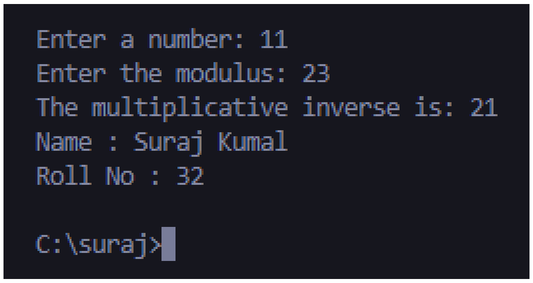
7. Program to check whether a number is multiplicative inverse of another number using brute force approach .

```
#include <iostream>
using namespace std;

int main() {
    int n, m = 1, mod;
    cout << "Enter a number: ";
    cin >> n;
    cout << "Enter the modulus: ";
    cin >> mod;

        // Using Bruteforce
    while (((n * m) % mod) != 1) {
        m++;
    }

    cout << "The multiplicative inverse is: " << m << endl;
    cout<<"Name : Suraj Kumal"<<endl<<"Roll No : 32"<<endl;
    return 0;
}
```

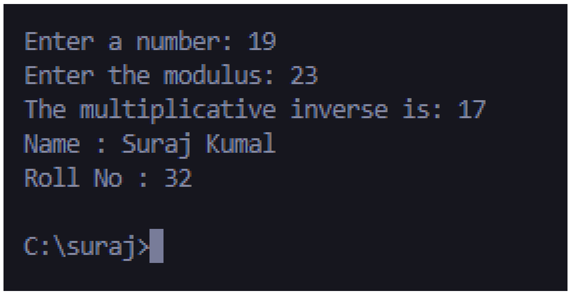
OUTPUT :



```
Enter a number: 7
Enter the modulus: 11
The multiplicative inverse is: 8
Name : Suraj Kumal
Roll No : 32

C:\suraj>
```



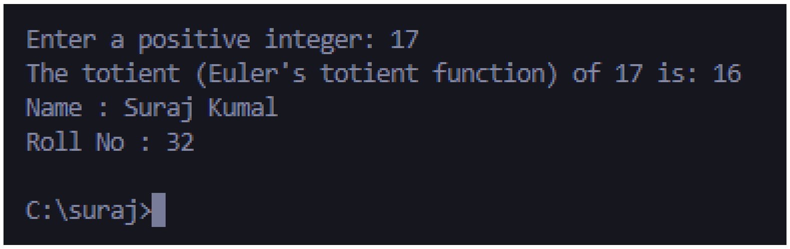
```
Enter a number: 11
Enter the modulus: 23
The multiplicative inverse is: 21
Name : Suraj Kumal
Roll No : 32

C:\suraj>
```



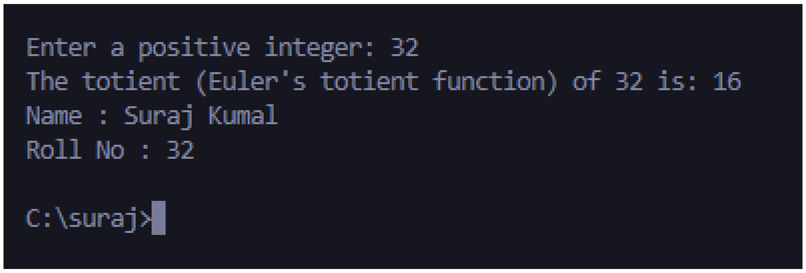
```
Enter a number: 19
Enter the modulus: 23
The multiplicative inverse is: 17
Name : Suraj Kumal
Roll No : 32

C:\suraj>
```

## 8. Program to compute totient of a number .

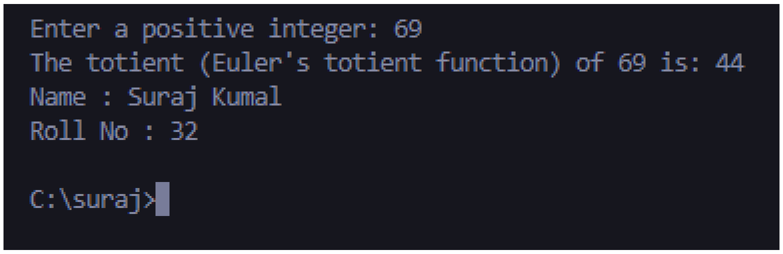
```
#include <iostream>
using namespace std;

int gcd(int a, int b) {
   while (b != 0) {
      int temp = b;
      b = a % b;
      a = temp;
   }
   return a;
}

int calculateTotient(int n) {
   int result = 1;  // Initialize the result as 1, since 1 is always coprime with any number.

   for (int i = 2; i < n; i++) {
      if (gcd(n, i) == 1) {
          result++;
      }
   }

   return result;
}

int main() {
   int n;

   cout << "Enter a positive integer: ";
   cin >> n;

   if (n > 0) {
      int totient = calculateTotient(n);
      cout << "The totient (Euler's totient function) of " << n << " is: " << totient << endl;
   } else {
      cout << "Please enter a positive integer." << endl;
   }
   cout<<"Name : Suraj Kumal"<<endl<<"Roll No : 32"<<endl;

   return 0;
}
```



```
Enter a positive integer: 17
The totient (Euler's totient function) of 17 is: 16
Name : Suraj Kumal
Roll No : 32

C:\suraj>
```



```
Enter a positive integer: 32
The totient (Euler's totient function) of 32 is: 16
Name : Suraj Kumal
Roll No : 32

C:\suraj>
```

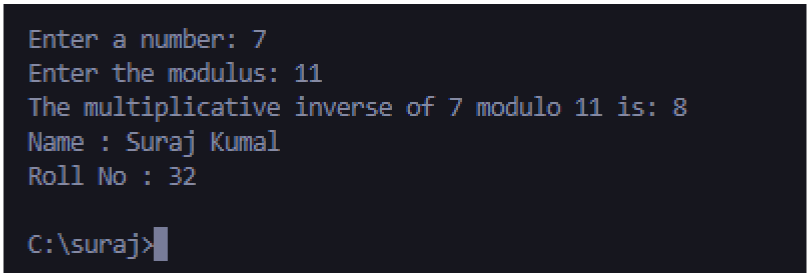


```
Enter a positive integer: 69
The totient (Euler's totient function) of 69 is: 44
Name : Suraj Kumal
Roll No : 32

C:\suraj>
```

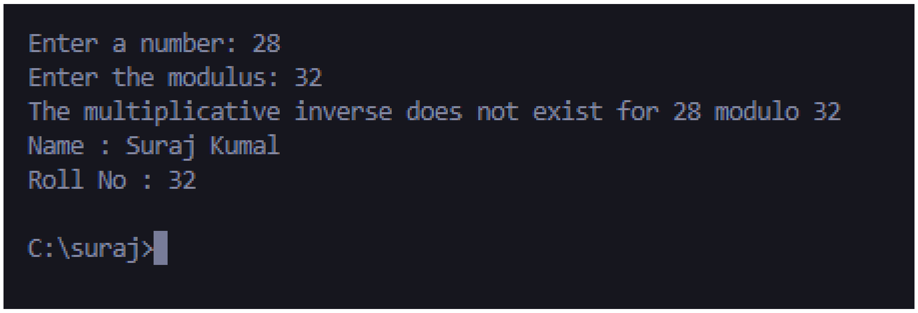
## 9. Program to compute multiplicative inverse of an integer .

```
#include <iostream>
using namespace std;

int extendedGCD(int a, int b, int &x, int &y) {
   if (a == 0) {
      x = 0;
      y = 1;
      return b;
   }

   int x1, y1;
   int gcd = extendedGCD(b % a, a, x1, y1);

   x = y1 - (b / a) * x1;
   y = x1;

   return gcd;
}

int multiplicativeInverse(int a, int modulus) {
   int x, y;
   int gcd = extendedGCD(a, modulus, x, y);

   if (gcd != 1) {
      return -1;
   } else {
      return (x % modulus + modulus) % modulus;
   }
}
```

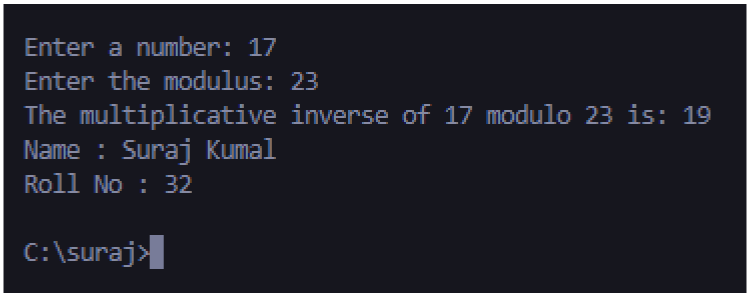
```
int main() {
    int a, modulus;

    cout << "Enter a number: ";
    cin >> a;

    cout << "Enter the modulus: ";
    cin >> modulus;

    int inverse = multiplicativeInverse(a, modulus);

    if (inverse != -1) {
        cout << "The multiplicative inverse of " << a << " modulo " << modulus << " is: " << inverse << endl;
    } else {
        cout << "The multiplicative inverse does not exist for " << a << " modulo " << modulus << endl;
    }
    cout<<"Name : Suraj Kumal"<<endl<<"Roll No : 32"<<endl;
    return 0;
}
```



```
Enter a number: 7
Enter the modulus: 11
The multiplicative inverse of 7 modulo 11 is: 8
Name : Suraj Kumal
Roll No : 32

C:\suraj>
```



```
Enter a number: 28
Enter the modulus: 32
The multiplicative inverse does not exist for 28 modulo 32
Name : Suraj Kumal
Roll No : 32

C:\suraj>
```

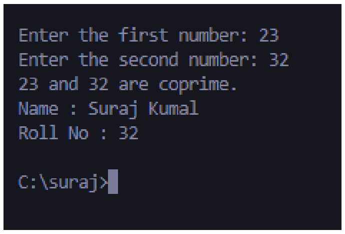


```
Enter a number: 17
Enter the modulus: 23
The multiplicative inverse of 17 modulo 23 is: 19
Name : Suraj Kumal
Roll No : 32

C:\suraj>
```

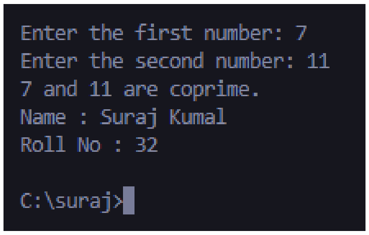
## 10. Program to check whether two numbers are co prime or not .

```
#include <iostream>
using namespace std;
// Function to calculate the GCD using the Euclidean algorithm
int gcd(int a, int b) {
   if (b == 0) {
      return a;
   }
   return gcd(b, a % b);
}

int main() {
   int num1, num2;
   // Input two numbers from the user
   cout << "Enter the first number: ";
   cin >> num1;
   cout << "Enter the second number: ";
   cin >> num2;
   // Calculate the GCD of the two numbers
   int greatestCommonDivisor = gcd(num1, num2);
   // Check if the GCD is 1 (numbers are coprime)
   if (greatestCommonDivisor == 1) {
      cout << num1 << " and " << num2 << " are coprime." << endl;
   } else {
      cout << num1 << " and " << num2 << " are not coprime." << endl;
   }
   cout<<"Name : Suraj Kumal"<<endl<<"Roll No : 32"<<endl;
   return 0;
}
```

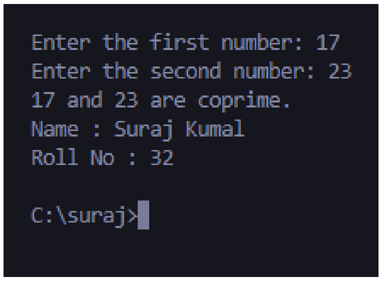
OUTPUT :



```
Enter the first number: 23
Enter the second number: 32
23 and 32 are coprime.
Name : Suraj Kumal
Roll No : 32

C:\suraj>
```



```
Enter the first number: 7
Enter the second number: 11
7 and 11 are coprime.
Name : Suraj Kumal
Roll No : 32

C:\suraj>
```

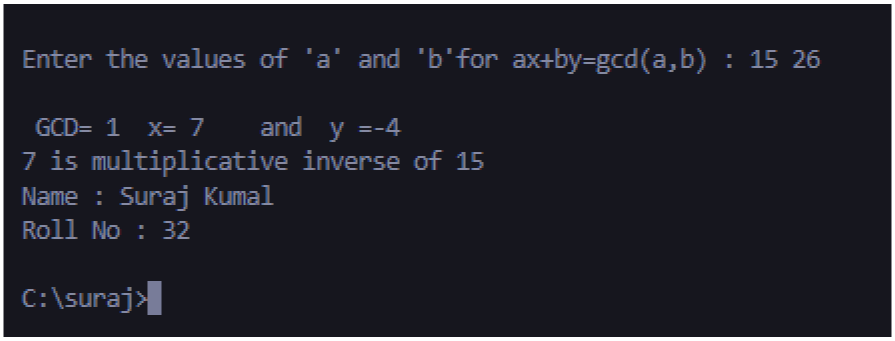


```
Enter the first number: 17
Enter the second number: 23
17 and 23 are coprime.
Name : Suraj Kumal
Roll No : 32

C:\suraj>
```

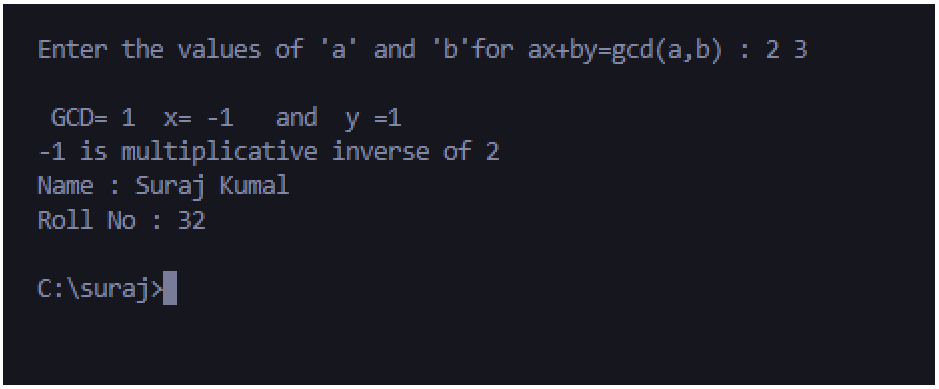
## 11. Program to implement Extended Euclidean algorithm.

```
#include<iostream>
using namespace std;
void xEuclidean(int a, int b,int &gcd,int &x_value, int &y_value)
{
   int old_x = 1, x = 0;
   int old_y = 0, y = 1;
   int q, r;
   while (b!=0){
      q = a / b;//quotient
      r = a % b;//remainder
      int temp = x;
      x = old_x - q * x;
      old_x = temp;
      temp = y;
      y = old_y - q * y;
      old_y = temp;
      a = b;
      b = r;
   }
   x_value=old_x;
   y_value=old_y;
   gcd=a;
}
int main(){
          int a,b,x,y,gcd;
          cout<<"\nEnter the values of 'a' and 'b'for ax+by=gcd(a,b) : ";
          cin>>a>>b;
          xEuclidean(a,b,gcd,x,y);
          cout<<"\n GCD= "<<gcd<<"\t x= "<<x<<"\t and  y ="<<y<<endl;
          cout<<x<<" is multiplicative inverse of "<<a<<endl;
          cout<<"Name : Suraj Kumal"<<endl<<"Roll No : 32"<<endl;
          return 0;
}
```

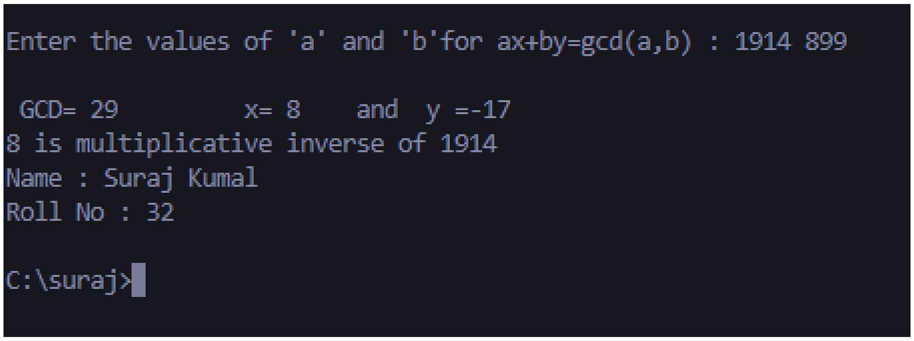
OUTPUT :



```
Enter the values of 'a' and 'b'for ax+by=gcd(a,b) : 15 26

 GCD= 1  x= 7    and  y =-4
7 is multiplicative inverse of 15
Name : Suraj Kumal
Roll No : 32

C:\suraj>
```



```
Enter the values of 'a' and 'b'for ax+by=gcd(a,b) : 2 3

 GCD= 1  x= -1   and  y =1
-1 is multiplicative inverse of 2
Name : Suraj Kumal
Roll No : 32

C:\suraj>
```

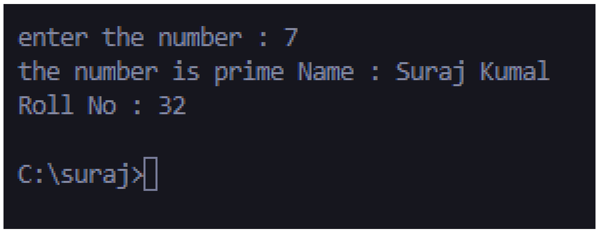


```
Enter the values of 'a' and 'b'for ax+by=gcd(a,b) : 1914 899

 GCD= 29         x= 8    and  y =-17
8 is multiplicative inverse of 1914
Name : Suraj Kumal
Roll No : 32

C:\suraj>
```

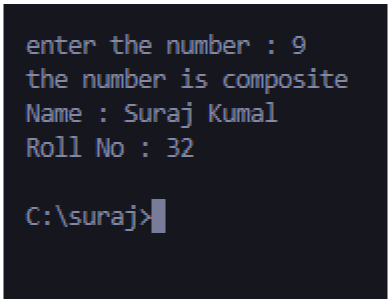
## 12. Program to check whether a given number is prime or not.

```
#include<iostream>
using namespace std;
int main(){
   int i, num ,count = 0;
   cout<<"enter the number : ";
   cin>>num;
   for(i=1;i<=num;i++){
     if(num%i==0){
      count ++;
     }
   }
   if(count==2){
      cout<<"the number is prime "<<endl;
   }
   else{
      cout<<"the number is composite"<<endl;
   }
   cout<<"Name : Suraj Kumal"<<endl<<"Roll No : 32"<<endl;
}
```

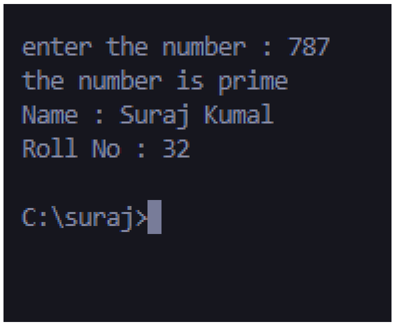
OUTPUT :



```
enter the number : 7
the number is prime Name : Suraj Kumal
Roll No : 32

C:\suraj>
```



```
enter the number : 9
the number is composite
Name : Suraj Kumal
Roll No : 32

C:\suraj>
```



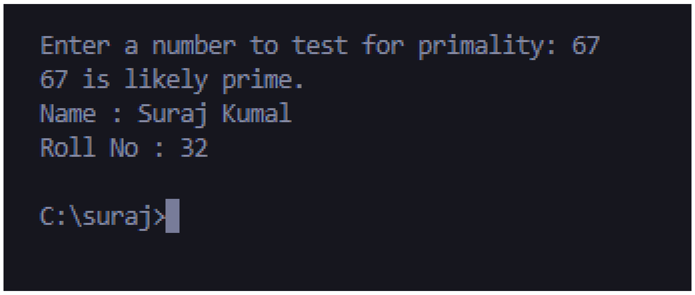
```
enter the number : 787
the number is prime
Name : Suraj Kumal
Roll No : 32

C:\suraj>
```

## 13. Program to perform primarity checking using  Rabin Miller algorithm.

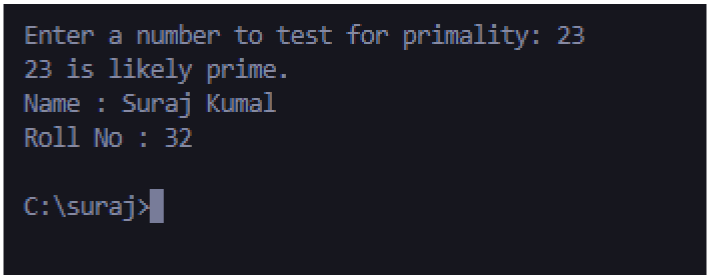
```
#include <iostream>
#include <cmath>
#include <cstdlib>
#include <ctime>

using namespace std;

// Modular exponentiation function (to compute a^b mod n)
long long mod_pow(long long base, long long exp, long long mod) {
  long long result = 1;
  while (exp > 0) {
    if (exp % 2 == 1) {
      result = (result * base) % mod;
    }
    base = (base * base) % mod;
    exp /= 2;
  }
  return result;
}

// Rabin-Miller primality test
bool is_prime_rabin_miller(long long n, int k) {
  if (n <= 1 || n == 4) {
    return false;
  }
  if (n <= 3) {
    return true;
  }

  // Find d and r such that n-1 = 2^r * d, where d is odd
  long long d = n - 1;
  int r = 0;
  while (d % 2 == 0) {
    d /= 2;
    r++;
  }

  // Witness loop
  for (int i = 0; i < k; i++) {
    long long a = 2 + rand() % (n - 3);
    long long x = mod_pow(a, d, n);
```

```
        if (x == 1 || x == n - 1) {
            continue;
        }
        bool is_composite = true;
        for (int j = 0; j < r; j++) { // Corrected the loop condition here
            x = mod_pow(x, 2, n);
            if (x == n - 1) {
                is_composite = false;
                break;
            }
        }

        if (is_composite) {
            return false;
        }
    }
    return true;
}

int main() {
    srand(time(nullptr)); // Seed the random number generator with the current time
    long long n;
    int k = 20; // Set a fixed value for the number of iterations
    cout << "Enter a number to test for primality: ";
    cin >> n;
    if (is_prime_rabin_miller(n, k)) {
        cout << n << " is likely prime." << endl;
    } else {
        cout << n << " is composite." << endl;
    }
    cout<<"Name : Suraj Kumal"<<endl<<"Roll No : 32"<<endl;
    return 0;
}
```

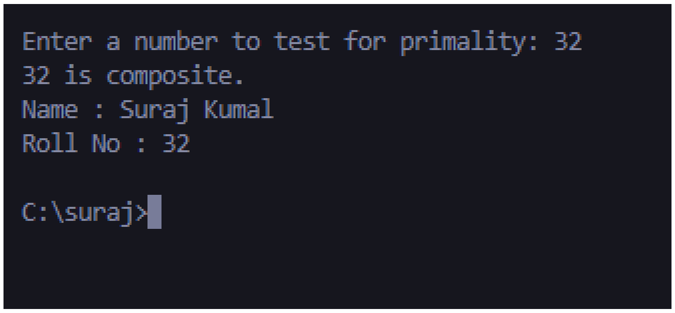
OUTPUT :



```
Enter a number to test for primality: 67
67 is likely prime.
Name : Suraj Kumal
Roll No : 32

C:\suraj>
```



```
Enter a number to test for primality: 23
23 is likely prime.
Name : Suraj Kumal
Roll No : 32

C:\suraj>
```

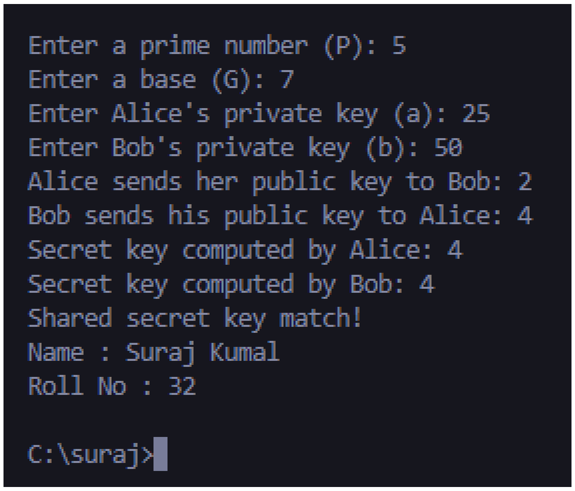


```
Enter a number to test for primality: 32
32 is composite.
Name : Suraj Kumal
Roll No : 32

C:\suraj>
```

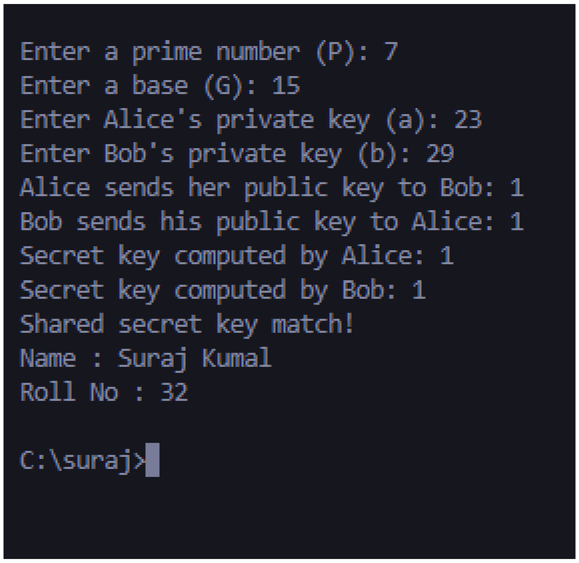
## 14.Program to implement Diffie-hellman algorithm.

```
#include <iostream>
#include <windows.h>
using namespace std;

// Function to calculate (base^exponent) % mod efficiently
long long modPow(long long base, long long exponent, long long mod) {
    long long result = 1;
    base %= mod;
    while (exponent > 0) {
        if (exponent % 2 == 1)
            result = (result * base) % mod;
        base = (base * base) % mod;
        exponent /= 2;
    }
    return result;
}

int main() {
    // Clear the console screen
    system("cls");

    long long prime, base, privateA, privateB, publicA, publicB, secretA, secretB;

    cout << "Enter a prime number (P): ";
    cin >> prime;

    cout << "Enter a base (G): ";
    cin >> base;

    cout << "Enter Alice's private key (a): ";
    cin >> privateA;

    cout << "Enter Bob's private key (b): ";
    cin >> privateB;

    // Calculate public keys
    publicA = modPow(base, privateA, prime);
    publicB = modPow(base, privateB, prime);

    // Exchange public keys (In a real-world scenario, this would be done over a secure channel)
    cout << "Alice sends her public key to Bob: " << publicA << endl;
    cout << "Bob sends his public key to Alice: " << publicB << endl;

    // Calculate shared secrets
    secretA = modPow(publicB, privateA, prime);
    secretB = modPow(publicA, privateB, prime);

    cout << "Secret key computed by Alice: " << secretA << endl;
    cout << "Secret key computed by Bob: " << secretB << endl;

    // Verify that the shared secrets are equal (they should be)
    if (secretA == secretB) {
        cout << "Shared secret key match! " << endl;
    } else {
        cout << "Shared secrets key do not match. Error in the key exchange." << endl;
    }
    cout<<"Name : Suraj Kumal"<<endl<<"Roll No : 32"<<endl;

    return 0;
}
```

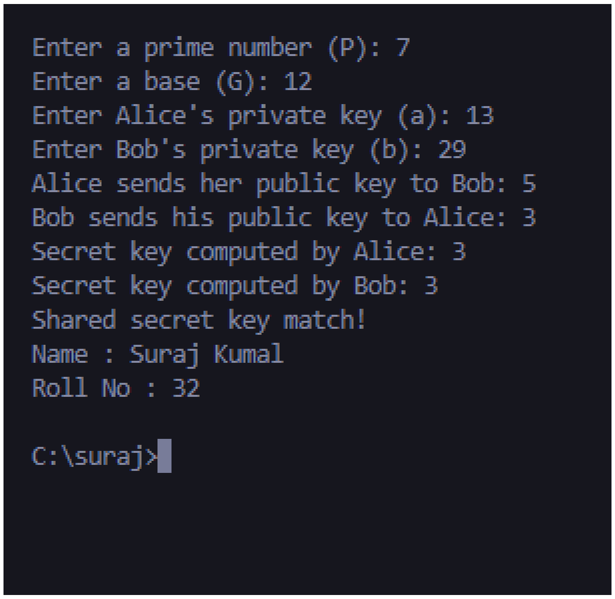
## OUTPUT :



```
Enter a prime number (P): 5
Enter a base (G): 7
Enter Alice's private key (a): 25
Enter Bob's private key (b): 50
Alice sends her public key to Bob: 2
Bob sends his public key to Alice: 4
Secret key computed by Alice: 4
Secret key computed by Bob: 4
Shared secret key match!
Name : Suraj Kumal
Roll No : 32

C:\suraj>
```



```
Enter a prime number (P): 7
Enter a base (G): 15
Enter Alice's private key (a): 23
Enter Bob's private key (b): 29
Alice sends her public key to Bob: 1
Bob sends his public key to Alice: 1
Secret key computed by Alice: 1
Secret key computed by Bob: 1
Shared secret key match!
Name : Suraj Kumal
Roll No : 32

C:\suraj>
```

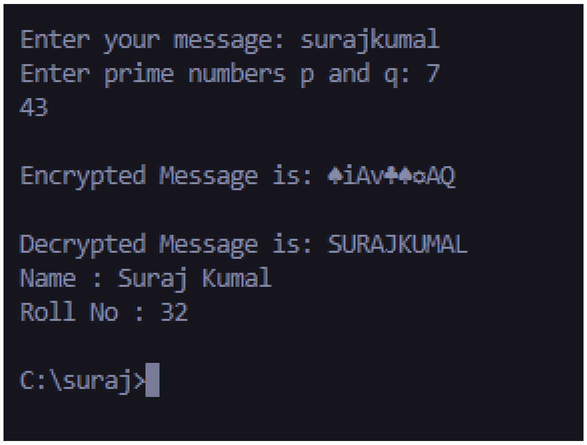


```
Enter a prime number (P): 7
Enter a base (G): 12
Enter Alice's private key (a): 13
Enter Bob's private key (b): 29
Alice sends her public key to Bob: 5
Bob sends his public key to Alice: 3
Secret key computed by Alice: 3
Secret key computed by Bob: 3
Shared secret key match!
Name : Suraj Kumal
Roll No : 32

C:\suraj>
```

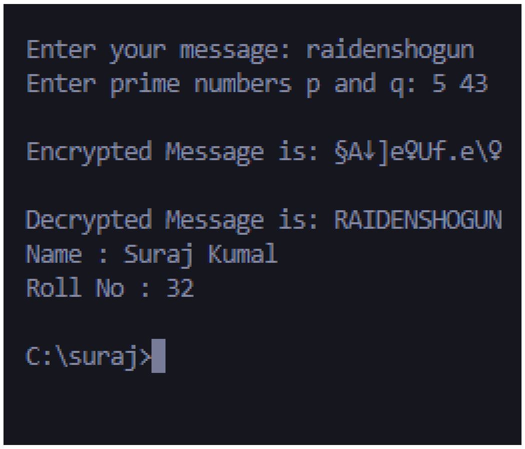
15. Program to implement key exchange and encryption decryption using RSA algorithm.

```
#include <iostream>
#include <string>

using namespace std;

int totient(int n) {
    int result = n;

    for (int p = 2; p * p <= n; ++p) {
        if (n % p == 0) {
            while (n % p == 0) {
                n /= p;
            }
            result -= result / p;
        }
    }

    if (n > 1) {
        result -= result / n;
    }

    return result;
}

int ext_euc(int a, int b, int &x, int &y) {
    if (b == 0) {
        x = 1;
        y = 0;
        return a;
    }

    int x1, y1;
    int gcd = ext_euc(b, a % b, x1, y1);
    x = y1;
    y = x1 - (a / b) * y1;
    return gcd;
}

int modulo(int base, int exponent, int mod) {
    int x = 1;
    int y = base;

    while (exponent > 0) {
        if (exponent % 2 == 1) {
            x = (x * y) % mod;
        }
        y = (y * y) % mod;
        exponent = exponent / 2;
    }

    if ((x % mod) < 0) {
        return (mod + (x % mod));
    } else {
        return x % mod;
    }
}
int main() {
    int cipher[50];
    int n, p, q, t, x, y, d = 1, e = 2;
    string message, encrypt, decrypt;

    cout << "Enter your message: ";
    getline(cin, message);

    for (char &c : message) {
            c = toupper(c) - 65;
    }

    cout << "Enter prime numbers p and q: ";
    cin >> p >> q;

    n = p * q;
    t = totient(n);

    while (ext_euc(e, t, x, y) != 1) {
        e++;
     }

    while (((e * d) % t) != 1) {
        d++;
    }

    cout << "\nEncrypted Message is: ";

    for (size_t i = 0; i < message.length(); i++) {
        cipher[i] = modulo(message[i], e, n);
        encrypt.push_back(static_cast<char>((modulo(message[i], e, n) + 65) % 128));
        cout << encrypt[i];
    }
    cout << "\n";
    cout << "\nDecrypted Message is: ";
    for (size_t i = 0; i < encrypt.length(); i++) {
        decrypt.push_back(static_cast<char>((modulo(cipher[i], d, n) + 65)));
        cout << decrypt[i];
    }
    cout << "\n";
    cout<<"Name : Suraj Kumal"<<endl<<"Roll No : 32"<<endl;
    return 0;
}
```

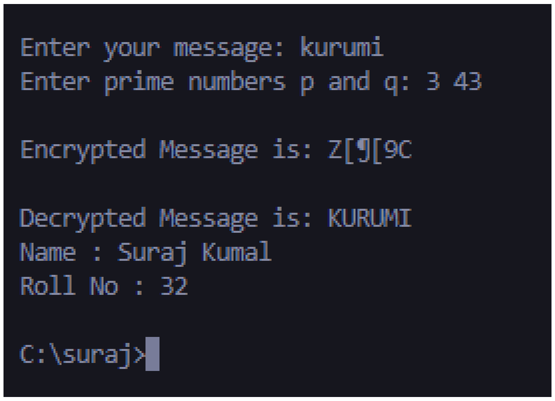
OUTPUT :



```
Enter your message: surajkumal
Enter prime numbers p and q: 7
43

Encrypted Message is: ♠iAv♣♠☼AQ

Decrypted Message is: SURAJKUMAL
Name : Suraj Kumal
Roll No : 32

C:\suraj>
```



```
Enter your message: raidenshogun
Enter prime numbers p and q: 5 43

Encrypted Message is: §A↓]e♀Uf.e\♀

Decrypted Message is: RAIDENSHOGUN
Name : Suraj Kumal
Roll No : 32

C:\suraj>
```



```
Enter your message: kurumi
Enter prime numbers p and q: 3 43

Encrypted Message is: Z[¶[9C

Decrypted Message is: KURUMI
Name : Suraj Kumal
Roll No : 32

C:\suraj>
```