CS18713 **SECURITY PRACTICES LABORATORY**

NUMBER THEORY
IMPLEMENTATION OF EUCLIDEAN ALGORITHM TO FIND THE GCD BETWEEN TWO NUMBERS

EX.NO. : 1a
DATE :

AIM:

To implement Euclidean Algorithm to find the gcd between two numbers

ALGORITHM :

1.Start
2. Define a function by taking two variables as parameters
3. Repeat steps 4 to 7 until a and b are not equal to 0
4. Find the modulo value of a and b
5. Put a=b and b = modulo value
6. If b equals 0 return a
7. Otherwise go to step 2 by passing new values of a and b
8. Stop

SOURCE CODE :

```
#include<stdio.h>
int euclid(int a ,int b){
while(a!=0 && b!=0){
   int  c = a%b;
   a=b;
   b=c;
   if(b==0)
   {
     return a;
   }
   else{
   euclid(a,b);
}}
}
void main(){
int a,b,gcd;
printf("Enter the values of a:");
scanf("%d",&a);
printf("Enter the value of b:");
```

```
scanf("%d",&b);
gcd = euclid(a,b);
printf("The GCD of the two numbers using Euclid algorithm is %d\n",gcd);
}
```

OUTPUT :

```
Enter the values of a:192
Enter the value of b:270
The GCD of the two numbers using Euclid algorithm is 6
```

RESULT :

Thus the euclidean algorithm to find the gcd of two numbers has been executed successfully

CS18713 **SECURITY PRACTICES LABORATORY**

NUMBER THEORY
IMPLEMENTATION OF EXTENDED EUCLIDEAN ALGORITHM TO FIND THE INVERSE MODULO

EX.NO. : 1b
DATE :

AIM:

To implement Extended Euclidean Algorithm to find the inverse modulo

ALGORITHM :

1.Start
2. Define a function to find the inverse modulo using extended euclids algorithm
3. Check if a is equal to 0
4. If a =0 then assign the value of the pointer x to be 0 and pointer y to be 1
5.Return the value of b
6. Otherwise call this function recursively by passing the modulo of a and b and the values of _x and _y
7.Update the values of the pointes and return the gcd value
8. Stop

SOURCE CODE :

```
#include <stdio.h>


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

   int _x, _y;
   int gcd = extended_gcd(b % a, a, &_x, &_y);

   *x = _y - (b/a) * _x;
   *y = _x;
```

```
    return gcd;
}

int main()
{
    int a,b;
    printf("Enter a:");
    scanf("%d",&a);
    printf("Enter b:");
    scanf("%d",&b);

    int x, y;

    printf("The GCD is %d\n", extended_gcd(a, b, &x, &y));
    printf("The inverse modulo value is = %d",x);

    return 0;
}
```

OUTPUT :

```
Enter a:15
Enter b:26
The GCD is 1
The inverse modulo value is = 7
```

RESULT :

Thus the extended euclidean algorithm to find the inverse modulo has been executed successfully.

CS18713 **SECURITY PRACTICES LABORATORY**

NUMBER THEORY
IMPLEMENTATION OF EULER TOTIENT FUNCTION

EX.NO. : 1c
DATE :

AIM:

To write a program to implement Euler Totient Function

ALGORITHM :

1. Start
2. Check if n is prime or not
3. If the number is prime then the Euler totient function will be n-1
4. If the number is composite then find the prime factors of the number
5. If the factors have exponents then the Euler totient function is $p^a - p^{a-1}$ * Eulter totient function of the non exponent prime factor
6. Stop

SOURCE CODE :

```
#include<stdio.h>
int gcd(int a,int b){
 if(a==0){
   return b;
 }
 return gcd(b%a,a);
}
int phi(unsigned int n){
 unsigned int result = 1;
int i;
 for( i=2;i<n;i++){
   if(gcd(i,n)==1){
       result++;
    }
 }
 return result;
}
int main(){
int n;
printf("Enter a number");
```

```
scanf("%d",&n);
printf("phi(%d) = %d\n",n,phi(n));
return 0;
}
```

OUTPUT :

```
Enter a number80
phi(80) = 32
```

RESULT :

Thus the program to implement euler totient function is executed successfully

CS18713 **SECURITY PRACTICES LABORATORY**

NUMBER THEORY
IMPLEMENTATION OF MILLER RABIN PRIMALITY CHECK

EX.NO. : 1d
DATE :

AIM:

To write a program to implement Miller Rabin Primality Check

ALGORITHM :

1..Start
2.Find integers k,q,d with K>0, q odd, so that (n-1 = 2^k * q);
3. Select a random integer a,1<a<n-1;
4. if a^q mod n = 1 then return prime;
5. for j = 0 to k-1 do;
6. if a^2jq mod n = n-1 then return prime;
7. return not prime
8. Stop

SOURCE CODE :

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
long long mulmod(long long a,long long b,long long mod){
long long x =0,y=a%mod;
while(b>0){
 if(b%2==1){
  x = (x+y)%mod;
 }
 y=(y*2)%mod;
 b/=2;
}
return x%mod;
}
long long modulo(long long base,long long exponent,long long mod){
long long x=1;
long long y=base;
while(exponent>0){
```

```
 if(exponent %2 ==1){
   x=(x*y)%mod;
 y=(y*y)%mod;
 exponent = exponent/2;
}
return x%mod;
}}
int miller(long long p,int iteration){
int i;
long long s;
if(p<2){
return 0;
}
if(p!=2 && p%2==0){
return 0;
}
s = p-1;
while(s%2==0){
s/=2;
}
for(i=0;i<iteration;i++){
long long a = rand()%(p-1)+1,temp=s;
long long mod = modulo(a,temp,p);
while(temp!=p-1 && temp%2==0){
return 0;
}
}
return 1;
}

main(){
int iteration =5;
long long num ;
printf("Enter a number to test:");
scanf("%lld",&num);
if(miller(num,iteration)){
printf("The number is prime");
}
else{
printf("Number is not prime");
```

```
}
return 0;
}
```

OUTPUT :

```
Enter a number to test:1729
The number is prime
```

RESULT :

Thus the program to implement Miller Rabin Primality Check is executed successfully

CS18713 **SECURITY PRACTICES LABORATORY**

SUBSTITUTION CIPHERS
IMPLEMENTATION OF CAESAR CIPHER

EX.NO. : 2a
DATE :

AIM:

To write a program to implement Caesar Cipher

ALGORITHM :

1. Start
2. Take the message and the key as input from the user
3. Add each character of the message with the key by using the ascii value of the characters
4. Repeat the same step for Upper case alphabets also.
5.Print the cipher text
6. For Decryption Process subtract the value of the key from the acscii value of the cipher text
7. Stop

SOURCE CODE :

ENCRYPTION:

```
#include<stdio.h>
int main(){
char message[100],ch;
int i,key;
printf("Enter the message to encrypt");
gets(message);
printf("Enter Key");
scanf("%d",&key);
for(i=0;message[i]!='\0';++i){
ch = message[i];
if(ch>='a'&&ch<='z'){
ch=ch+key;
if(ch>'z'){
ch = ch-'z'+'a'-1;
}
message[i]=ch;
}
```

```
else if(ch>='A' && ch<='Z'){
ch=ch+key;
if(ch>'Z'){
ch = ch-'Z'+'A'-1;
}
message[i]=ch;
}
}
printf("Encrypted message using Caeser cipher is %s",message);
return 0;
}
```

OUTPUT :

```
Enter the message to encrypt:abcd
Enter Key4
Encrypted message using Caeser cipher is efgh
```

DECRYPTION:

```
#include<stdio.h>
int main(){
char message[100],ch;
int i,key;
printf("Enter the message to decrypt:");
gets(message);
printf("Enter Key: ");
scanf("%d",&key);
for(i=0;message[i]!='\0';++i){
ch = message[i];
if(ch>='a'&&ch<='z'){
ch=ch-key;
if(ch>'z'){
ch = ch-'z'+'a'-1;
}
message[i]=ch;
}
else if(ch>='A' && ch<='Z'){
ch=ch-key;
if(ch>'Z'){
ch = ch-'Z'+'A'-1;
```

```
}
message[i]=ch;
}
}
printf("Decrypted message using Caeser cipher is %s",message);
return 0;
}
```

OUTPUT:

```
Enter the message to decrypt: efgh
Enter Key: 4
Decrypted message using Caeser cipher is abcd
```

RESULT :

Thus the program to implement Caesar Cipher is executed successfully.

CS18713 **SECURITY PRACTICES LABORATORY**

SUBSTITUTION CIPHERS
IMPLEMENTATION OF PLAYFAIR CIPHER

EX.NO. : 2b
DATE :

AIM:

To write a program to implement PlayFair Cipher

ALGORITHM :

1. Start
2.Take the plain text and key as input.
3. Divide the text into letters of two letters and use fillers such as x
4. Construct a table with the key and fill the remaining spaces of the table with the remaining alphabets
5. Find the cipher text by checking the row and the column of the plain text in the table.
6. If they are in different rows then find the rowwise and columnwise intersection and consider it to be the cioher text.
7. If they are in the same row then take the next characters to be the cipher text.
8. Print the cipher text
9. Stop

SOURCE CODE :

```
#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;
            }
}

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;

   int n; // number of keys
   cout << "Enter number of keys : "; cin >> n;

   string key[n];
   for(int i=0; i<n; i++)
   {
      cout<< "\nEnter key " << i+1 << " : " << key[i];
      cin >> key[i];

      generateMatrix(key[i]);

      cout << "Key " << i+1 << " 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 \t\t: " << plaintext << endl;

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

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

      string decryptmsg = Decrypt(ciphertext);
      cout<< "Decrypted Message \t: " << decryptmsg << endl;
   }
}
```

OUTPUT :

```
Enter message : india
Enter number of keys : 1

Enter key 1 : monarchy
Key 1 Matrix:
m o n a r
c h y b d
e f g i k
l p q s t
u v w x z
Actual Message        : india
Formatted Message     : indiax
Encrypted Message     : gabkba
Decrypted Message     : indiax
```

RESULT :

Thus the program to implement PlayFair Cipher is executed successfully.

CS18713 **SECURITY PRACTICES LABORATORY**

SUBSTITUTION CIPHERS
IMPLEMENTATION OF HILL CIPHER

EX.NO. : 2c
DATE :

AIM:

To write a program to implement Hill Cipher

ALGORITHM :

1. Start
2. Convert the given key into a matrix format by taking the corresponding numbers
3. Then convert the given plain text into a matrix format
4. Perform Matrix multiplication mod 26 of the matrices obtained in step 2 and 3
5. The convert the obtained number into a text to obtain the cipher text
6. To decrypt the cipher text multiply it with the inverse of the key matrix
7. Stop

SOURCE CODE :

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

using namespace std;

float encrypt[3][1], decrypt[3][1], a[3][3], b[3][3], mes[3][1], c[3][3];

void encryption();
void decryption();
void getKeyMessage();
void inverse();

int main() {
        getKeyMessage();
        encryption();
        decryption();
}
```

```
void encryption() {
	int i, j, k;

	for(i = 0; i < 3; i++)
		for(j = 0; j < 1; j++)
			for(k = 0; k < 3; k++)
				encrypt[i][j] = encrypt[i][j] + a[i][k] * mes[k][j];

	cout<<"\ncipher text: ";
	for(i = 0; i < 3; i++)
		cout<<(char)(fmod(encrypt[i][0], 26) + 97);
}

void decryption() {
	int i, j, k;

	inverse();

	for(i = 0; i < 3; i++)
		for(j = 0; j < 1; j++)
			for(k = 0; k < 3; k++)
				decrypt[i][j] = decrypt[i][j] + b[i][k] * encrypt[k][j];

	cout<<"\n De-ciphered text: ";
	for(i = 0; i < 3; i++)
		cout<<(char)(fmod(decrypt[i][0], 26) + 97);

	cout<<"\n";
}

void getKeyMessage() {
	int i, j;
	char msg[3];

	cout<<"\nkey [3x3]:\n";

	for(i = 0; i < 3; i++)
		for(j = 0; j < 3; j++) {
			cin>>a[i][j];
			c[i][j] = a[i][j];
```

```
		}

	cout<<"\nplain text[3 character]: ";
	cin>>msg;

	for(i = 0; i < 3; i++)
		mes[i][0] = msg[i] - 97;
}

void inverse() {
	int i, j, k;
	float p, q;

	for(i = 0; i < 3; i++)
		for(j = 0; j < 3; j++) {
			if(i == j)
				b[i][j]=1;
			else
				b[i][j]=0;
		}
	for(k = 0; k < 3; k++) {
		for(i = 0; i < 3; i++) {
			p = c[i][k];
			q = c[k][k];

			for(j = 0; j < 3; j++) {
				if(i != k) {
					c[i][j] = c[i][j]*q - p*c[k][j];
					b[i][j] = b[i][j]*q - p*b[k][j];
				}
			}
		}
	}

	for(i = 0; i < 3; i++)
		for(j = 0; j < 3; j++)
			b[i][j] = b[i][j] / c[i][i];

}
```

OUTPUT :

key [3x3]:
6 24 1
13 16 10
20 17 15

Plain text[3 character]: act
Cipher text: poh
De-ciphered text: act

RESULT :

Thus the program to implement Hill Cipher is executed successfully.

CS18713 **SECURITY PRACTICES LABORATORY**

SUBSTITUTION CIPHERS
IMPLEMENTATION OF VIGENERE CIPHER

EX.NO. : 2d
DATE :

AIM:

To write a program to implement Vigenere Cipher

ALGORITHM :

1. Start
2. Take the plain text and the key as input from the user
3.Pad the key by repeating the characters so that the length of the key matches the length of the plain text.
4. Then add the ascii values of the plaintext and the key to get the cipher text.
5. For the decryption process subtract the key from the cipher text.
6. Stop

SOURCE CODE :

```
#include<stdio.h>
#include<string.h>

int main(){
   char msg[100];
   char key[100];
   printf("Enter the Plain text:");
   scanf("%s",msg);
   printf("Enter the key:");
   scanf("%s",key);
   int msgLen = strlen(msg), keyLen = strlen(key), i, j;

   char newKey[msgLen], encryptedMsg[msgLen], decryptedMsg[msgLen];

   for(i = 0, j = 0; i < msgLen; ++i, ++j){
      if(j == keyLen)
         j = 0;

      newKey[i] = key[j];
   }
```

```
    newKey[i] = '\0';

    //encryption
    for(i = 0; i < msgLen; ++i)
        encryptedMsg[i] = ((msg[i] + newKey[i]) % 26) + 'A';

    encryptedMsg[i] = '\0';

    //decryption
    for(i = 0; i < msgLen; ++i)
        decryptedMsg[i] = (((encryptedMsg[i] - newKey[i]) + 26) % 26) + 'A';

    decryptedMsg[i] = '\0';

    printf("Original Message: %s", msg);
    printf("\nKey: %s", key);
    printf("\nNew Generated Key: %s", newKey);
    printf("\nEncrypted Message: %s", encryptedMsg);
    printf("\nDecrypted Message: %s", decryptedMsg);

return 0;
}
```

OUTPUT :

```
Enter the Plain text:SVCE
Enter the key:CAT
Original Message: SVCE
Key: CAT
New Generated Key: CATC
Encrypted Message: UVVG
Decrypted Message: SVCE
```

RESULT :

Thus the program to implement Vigenere Cipher is executed successfully

CS18713 **SECURITY PRACTICES LABORATORY**

SUBSTITUTION CIPHERS
IMPLEMENTATION OF ONE TIME PAD CIPHER

EX.NO. : 2e
DATE :

AIM:

To write a program to implement One time pad Cipher

ALGORITHM :

1. Start
2. Take the plaintext as input
3.Convert the plaintext to upper case
4. Find the number corresponding to the text
5. Take the key as input
6. Convert the key also to upper case
7. Obtain the numerical key for the one time pad
8. Using this numerical key encrypt the plaintext.
9. Stop

SOURCE CODE :

ENCRYPTION:

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>
main()
{
int i,j,len1,len2,numstr[100],numkey[100],numcipher[100];
 char str[100],key[100],cipher[100];
 printf("Enter a string text to encrypt\n");
 gets(str);
 for(i=0,j=0;i<strlen(str);i++)
 {
  if(str[i]!=' ')
  {
   str[j]=toupper(str[i]);
   j++;
  }
 }
```

```
 str[j]='\0';
    for(i=0;i<strlen(str);i++)
    {
     numstr[i]=str[i]-'A';
    }
    printf("Enter key string of random text\n");
    gets(key);
 for(i=0,j=0;i<strlen(key);i++)
 {
  if(key[i]!=' ')
  {
   key[j]=toupper(key[i]);
   j++;
  }
 }
 key[j]='\0';

    for(i=0;i<strlen(key);i++)
    {
     numkey[i]=key[i]-'A';
    }

    for(i=0;i<strlen(str);i++)
    {
     numcipher[i]=numstr[i]+numkey[i];
    }
    for(i=0;i<strlen(str);i++)
    {
     if(numcipher[i]>25)
     {
      numcipher[i]=numcipher[i]-26;
     }
    }
    printf("One Time Pad Cipher text is\n");
    for(i=0;i<strlen(str);i++)
    {
      printf("%c",(numcipher[i]+'A'));
    }
    printf("\n");
}
```

OUTPUT :

Enter a string text to encrypt
SVCE
Enter key string of random text
COLLEGE
One Time Pad Cipher text is
UJNP

DECRYPTION:

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>

main()

{

 int i,j,len1,len2,numstr[100],numkey[100],numcipher[100];

 char str[100],key[100],cipher[100];

 printf("Enter an Encrypted string text to Decrypt\n");

 gets(str);

 for(i=0,j=0;i<strlen(str);i++)

 {

  if(str[i]!=' ')

  {

   str[j]=toupper(str[i]);

   j++;

  }

 }

str[j]='\0';

    for(i=0;i<strlen(str);i++)

    {numstr[i]=str[i]-'A'; }
```

```
   printf("Enter key string of random text\n");
   gets(key);
 for(i=0,j=0;i<strlen(key);i++)
 {
  if(key[i]!=' ')
  {
   key[j]=toupper(key[i]);
   j++;
  }
 }
 key[j]='\0';
  for(i=0;i<strlen(key);i++)
    {
     numkey[i]=key[i]-'A';
    }

   for(i=0;i<strlen(str);i++)
   {
    numcipher[i]=numstr[i]-numkey[i];//changed from + to - for decryption
    if(numcipher[i]<0)
    {
     numcipher[i]=numcipher[i]+26;//If cipher is negative we have to add 26
    }
    numcipher[i]=numcipher[i]%26;//To loop within 1 to 26 for alphabets from A-Z
   }
```

```
    printf("Decrypted One Time Pad Cipher text is\n");
    for(i=0;i<strlen(str);i++)
    {
      printf("%c",(numcipher[i]+'A'));
    }
    printf("\n");
}
```

OUTPUT:

Enter an Encrypted string text to Decrypt
UJNP
Enter key string of random text
COLLEGE
Decrypted One Time Pad Cipher text is
SVCE

RESULT :

Thus the program to implement One time pad Cipher is executed successfully

CS18713 **SECURITY PRACTICES LABORATORY**

TRANSPOSITIONAL CIPHERS
IMPLEMENTATION OF RAIL FENCE CIPHER

EX.NO. : 3a
DATE :

AIM:

To implement Rail fence cipher

ALGORITHM :

1.Start
2.Take the plain text as input
3.Write the plaintext alternatively in each row
4.To retrieve the cipher text read the characters in each row separately.
5.Stop

SOURCE CODE :

```
#include<stdio.h>

#include<string.h>

void main()
{
int i,j,k,l;
char a[20],c[20],d[20];
printf("\n\t\t RAIL FENCE TECHNIQUE");
printf("\n\nEnter the input string : ");
gets(a);
l=strlen(a);

for(i=0,j=0;i<l;i++)
{
if(i%2==0)
c[j++]=a[i];
}
for(i=0;i<l;i++)
{
if(i%2==1)
c[j++]=a[i];
```

```
}
c[j]='\0';
printf("\nCipher text after applying rail fence :");
printf("\n%s",c);

if(l%2==0)
k=l/2;
else
k=(l/2)+1;
for(i=0,j=0;i<k;i++)
{
d[j]=c[i];
j=j+2;
}
for(i=k,j=1;i<l;i++)
{
d[j]=c[i];
j=j+2;
}
d[l]='\0';
printf("\nText after decryption : ");
printf("%s",d);
}
```

OUTPUT :

```
RAIL FENCE TECHNIQUE

Enter the input string : cryptography
Cipher text after applying rail fence :
cytgahrporpy
Text after decryption : cryptography
```

RESULT :

Thus the Rail Fence cipher has been executed successfully.

CS18713 **SECURITY PRACTICES LABORATORY**

TRANSPOSITIONAL CIPHERS
IMPLEMENTATION OF ROW COLUMN TRANSPOSITION

EX.NO. : 3b
DATE :

AIM:

To implement Row Column transposition cipher

ALGORITHM :

1.Start
2.Take the plain text and key as input
3.The message is written out in rows of a fixed length.
4.Then read out again column by column, and the columns are chosen in a scrambled order.
5.Finally, the message is read off in columns, in the order specified by the keyword.
6.Stop

SOURCE CODE :

```
def split_len(seq, length):
   return [seq[i:i + length] for i in range(0, len(seq), length)]
def encode(key, plaintext):
   order = {
     int(val): num for num, val in enumerate(key)
   }
   ciphertext = ''
   for index in sorted(order.keys()):
       for part in split_len(plaintext, len(key)):
           try:ciphertext += part[order[index]]
           except IndexError:
              continue
   return ciphertext
print(encode('3214', 'CRYPTOGRAPHY'))
```

OUTPUT :

YGHROPCTAPRY

RESULT :

Thus the Row column transpositional cipher is executed successfully.

CS18713 **SECURITY PRACTICES LABORATORY**

IMPLEMENTATION OF SDES ALGORITHM

EX.NO. : 4
DATE :

AIM:

To write a program to implement SDES Algorithm

ALGORITHM :

1.Start
2. Define the permutation and inverse permutation tables
3.Define functions to perform shift operation,permutations and xor operations
4.Using the shifting and permutations functions generate 2 keys for the sdes algorithm
5.Define functions for binary to decimal and decimal to binary
6. Define the process of encryption and decryption by using the keys and the plain text and pass it into the p10 permutation first to get the two keys and then into p8 permutations for 2 rounds to get the cipher text
7. For Decryption use the keys in the reverse order on the cipher text to get back the plain text
8. Stop

SOURCE CODE :

```
p10 = [3,5,2,7,4,10,1,9,8,6]
p8 = [6,3,7,4,8,5,10,9]
p4 = [2,4,3,1]
IP = [2,6,3,1,4,8,5,7]
IP_inv = [4,1,3,5,7,2,8,6]
expas = [4,1,2,3,2,3,4,1]
s0 = [[1,0,3,2],[3,2,1,0],[0,2,1,3],[3,1,3,2]]
s1 = [[0,1,2,3],[2,0,1,3],[3,0,1,0],[2,1,0,3]]
key = [1,0,1,0,0,0,0,0,1,0]

def shift(l,r,n):
  return l[n:]+l[0:n],r[n:]+r[0:n]

def permutation(bitstirng,arr):
  return "".join(bitstirng[index-1] for index in arr)

def expansion(bitstring,expans):
    return "".join([bitstring[expans[i]-1] for i in range(len(expans))])
```

```
def xor(bitstring,key):
  ans=""
  for i in range(len(bitstring)):
    if(bitstring[i]==key[i]):
      ans+="0";
    else:
      ans+="1";
  return ans
def genrate_key(k):
  tmpkey=permutation(k,p10);
  l,r=tmpkey[0:5],tmpkey[5:];
  l,r=shift(l,r,1)
  k1=permutation(l+r,p8);
  l,r=shift(l,r,2)
  k2=permutation(l+r,p8);
  return k1[0:8],k2[0:8]

def bitTodec(str):
  dec={
    "00":0,
    "01":1,
    "10":2,
    "11":3
  }
  return dec.get(str);
def decTobin(str):
  dec={
    0:"00",
    1:"01",
    2:"10",
    3:"11"
  }
  return dec.get(str);
def givesboxoutput(bitstring,s
  row=bitTodec(bitstring[0]+bitstring[3])
  col=bitTodec(bitstring[1]+bitstring[2])
  return decTobin(s[row][col])

def fk(r1,k1):  #f(k) function
  mid=expansion(r1,expas)
```

```
  exor_out=xor(mid,k1);
  sbox_out1,sbox_out2=givesboxoutput(exor_out[0:4],s0),givesboxoutput(exor_out[4:],s1)
  return permutation(sbox_out1+sbox_out2,p4)


def process(plain,k1,k2):
  mid=permutation(plain,IP)
  l1,r1=mid[0:4],mid[4:]
  fkoutput=fk(r1,k1)
  l2,r2=r1,xor(fkoutput,l1)
  fkoutput=fk(r2,k2)
  l2,r2=xor(l2,fkoutput),r2
  return permutation(l2+r2,IP_inv)

if __name__=='__main__':
  k1,k2=genrate_key(".join(map(str,key)));
  pt = input("Enter plaintext enclosed within double quotes")
  ciphertext=process(pt,k1,k2); transformation(ENCRYTION)
  print("Cipher Text: ",ciphertext)
  plaintext=process(ciphertext,k2,k1)
  print("Plain Text: ",plaintext)
```

OUTPUT :

(‘Cipher text:01110111’)
(‘Plain text: 01110010’)

RESULT :

Thus the program to implement SDES Algorithm is executed successfully