VISHWAKARMA GOVERNMENT ENGINEERING COLLEGE, CHANDKHEDA-382424

Information Technology Department

Subject Name:	**3161606: Cryptography and Network Security**
Branch & Semester:	Information Technology, Sem-6
Enrollment No.	**190173116006**
Name:	**Gabani Parth Naranbhai**

Practical Index

Sr. No.	Aim of Practical	Date	Marks
1.	Create your own logic to perform encoding of a text message and also decode it to get original text message back.	26/12	9
2.	Implement a program to perform encryption and decryption using Caesar cipher algorithm.	26/12	10
3.	Implement a program to find plain text messages and key information corresponds to following cipher text messages using brute-force technique on Caesar cipher. • Qefpfpzxbpbozfmeboxidlofqej • TrvjviTzgyvizjNvrbRcxfizkyd • LbhNerFzneggbNggnpxPnrfrePvcure	3/1	9
4.	Implement a program to perform encryption and decryption using Monoalphabetic cipher algorithm	3/1	9
5.	Implement a program to find GCD using Euclidean algorithm & multiplicative inverse using Extended Euclidean algorithm.	9/1	8
6.	Implement a program that returns the value of Euler's totient function.	13/1	10
7.	Implement a program to perform encryption and decryption using Affine cipher algorithm.	20/2	10
8.	Implement a program to perform encryption and decryption using Playfair Cipher.	6/3	8
9.	Implement a program to perform encryption and decryption using Hill Cipher.	13/3	8
10.	Implement a program to perform encryption and decryption using Rail fence Cipher.	20/3	10
11.	Implement a program to perform encryption and decryption using Columnar transposition algorithm.	27/3	8
12.	Implement a program to perform encryption and decryption using DES Algorithm.	20/4	7
13.	Implement a program to perform encryption and decryption using RSA Algorithm.	17/4	8
14.	Implement a program to generate key using Diffie-Hellman key exchange algorithm and using that key perform encryption and decryption using Caesar cipher algorithm.	17/4	9

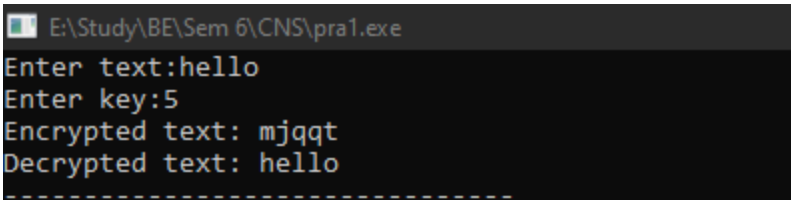
Practical 1

Aim: - Create your own logic to perform encoding of a text message and also decode it to get original text message back.

- **Code: -**

```
#include<stdio.h>
#define MAX 100

void main(){

	char ch,p[MAX],c[MAX],d[MAX];
	int key,i;

	printf("Enter text:");
	scanf("%s",&p);

	printf("Enter key:");
	scanf("%d",&key);

	for(i=0;i<strlen(p);i++){
		c[i] = p[i] + key;
	}

	printf("Encrypted text: %s\n",c);

	for(i=0;i<strlen(p);i++){
		d[i] = c[i] -key;
	}

	printf("Decrypted text: %s",d);
}
```

- **Output: -**


E:\Study\BE\Sem 6\CNS\pra1.exe
Enter text:hello
Enter key:5
Encrypted text: mjqqt
Decrypted text: hello

Practical 2

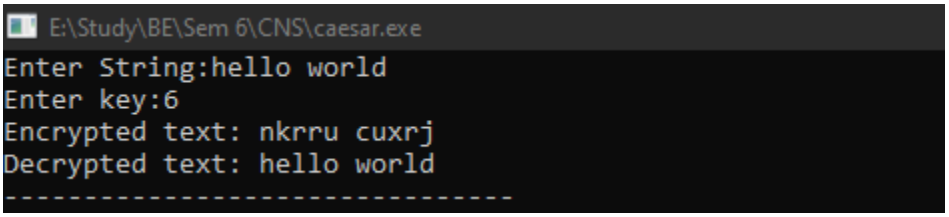
Aim: - Implement a program to perform encryption and decryption using Caesar cipher algorithm.

- **Code: -**

```
#include<stdio.h>
#define max 100

int n,ch;
char s[max],c[max],d[max];

void main(){
	int i;

	printf("Enter String:");
	scanf("%[^\n]",&s);

	printf("Enter key:");
	scanf("%d",&n);

	if(n>26){
		n = n%26;
	}

	for(i=0;i<strlen(s);i++){
		ch = s[i];
		if(ch>='a' && ch<='z'){
			ch = ch + n;
			if(ch>'z'){
				ch = ch - 'z' + 'a' - 1;

			}

			c[i] = ch;
		}
```

```
        else if(ch>='A' && ch<='Z'){
                ch = ch + n;
                if(ch>'Z'){
                        ch = ch - 'Z' + 'A' - 1;
                }
                c[i] = ch;
        }
        else{
                c[i] = ch;
        }
    }

    printf("Encrypted text: %s",&c);

    for(i=0;i<strlen(c);i++){
        ch = c[i];

        if(ch>='a' && ch<='z'){
                ch = ch - n;
                if(ch<'a'){
                        ch = ch + 'z' - 'a' + 1;
                }
                d[i] = ch;
        }
        else if(ch>='A' && ch<='Z'){
                ch = ch - n;
                if(ch<'A'){
                        ch = ch + 'Z' - 'A' + 1;
                }
                d[i] = ch;
        }
        else{
                d[i] = c[i];
        }
    }
    printf("\nDecrypted text: %s",&d);
}
```

- **Output: -**


E:\Study\BE\Sem 6\CNS\caesar.exe
Enter String:hello world
Enter key:6
Encrypted text: nkrru cuxrj
Decrypted text: hello world

Practical 3

Aim: - Implement a program to find plain text messages and key information corresponds to following cipher text messages using brute-force technique on Caesar cipher.

- Qefpfpzxbpbozfmeboxidlofqej
- TrvjviTzgyvizjNvrbRcxfizkyd
- LbhNerFzneggbNggnpxPnrfrePvcure

➢ **Code: -**

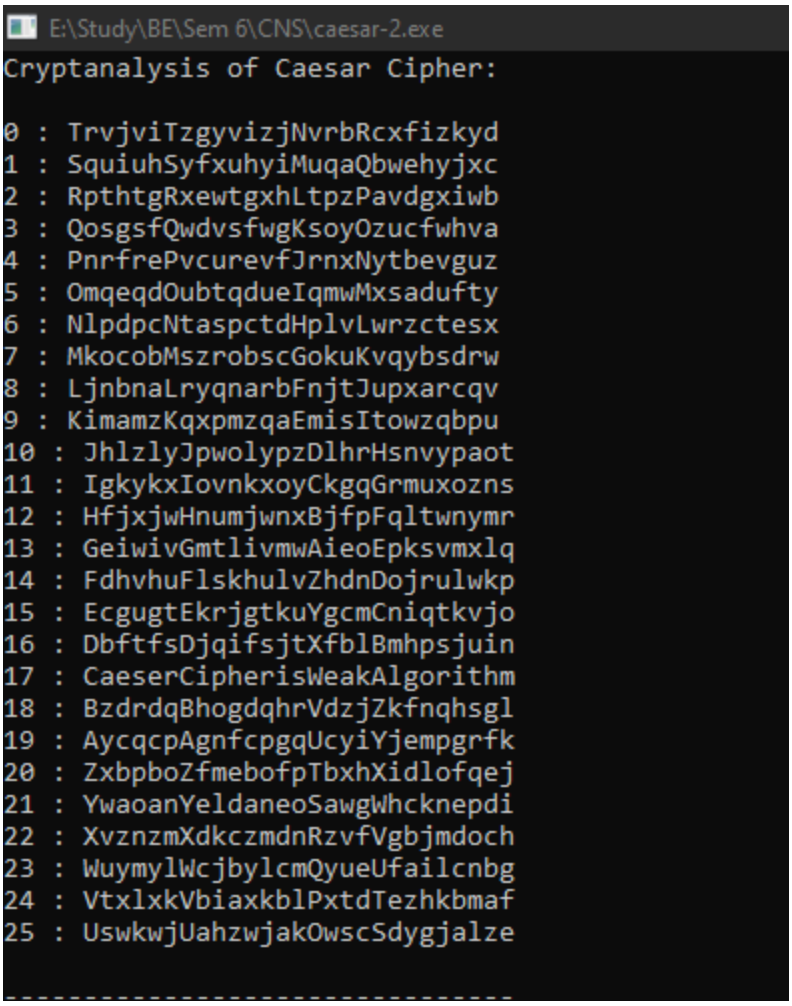
```
#include<stdio.h>
#define MAX 100

char c[MAX] = "LbhNerFzneggbNggnpxPnrfrePvcure",d[MAX];

void main(){

	int i,j,ch;

	printf("Cryptanalysis of Caesar Cipher:\n\n");

	for(i=0;i<=25;i++){
		for(j=0;j<strlen(c);j++){
		ch = c[j];

			if(ch>='a' && ch<='z'){
				ch = ch - i;
				if(ch<'a'){
					ch = ch + 26 ;
				}
				d[j] = ch;
			}
			else if(ch>='A' && ch<='Z'){
				ch = ch - i;
				if(ch<'A'){
```

```
                        ch = ch + 26 ;
                    }
                    d[j] = ch;
                }
                else{
                    d[j] = c[j];
                }
            }
            printf("%d : %s\n",i,d);
        }
}
```

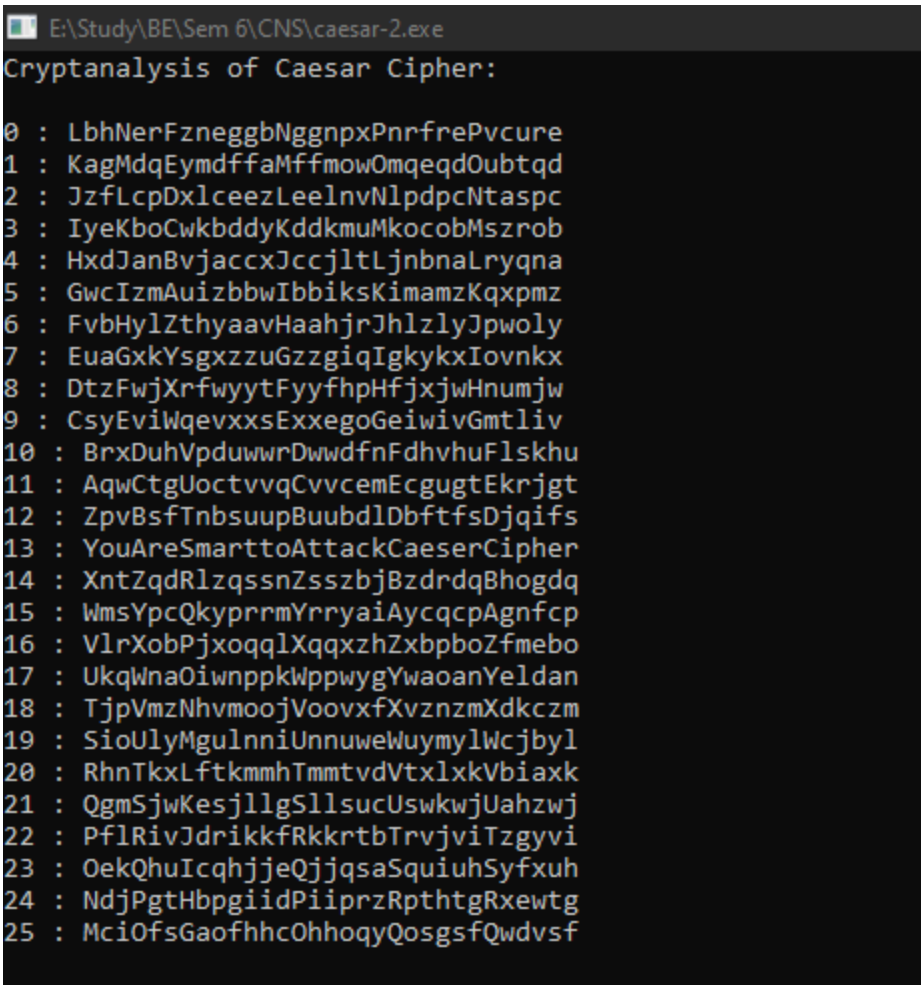
- **Output: -**

```
E:\Study\BE\Sem 6\CNS\caesar-2.exe
Cryptanalysis of Caesar Cipher:

0 : Qefpfpzxbpbozfmeboxidlofqej
1 : Pdeoeoywaoanyeldanwhcknepdi
2 : Ocdndnxvznzmxdkczmvgbjmdoch
3 : Nbcmcmwuymylwcjbylufailcnbg
4 : Mablblvtxlxkvbiaxktezhkbmaf
5 : Lzakakuswkwjuahzwjsdygjalze
6 : Kyzjzjtrvjvitzgyvircxfizkyd
7 : Jxyiyisquiuhsyfxuhqbwehyjxc
8 : Iwxhxhrpthtgrxewtgpavdgxiwb
9 : Hvwgwgqosgsfqwdvsfozucfwhva
10 : Guvfvfpnrfrepvcurenytbevguz
11 : Ftueueomqeqdoubtqdmxsadufty
12 : Estdtdnlpdpcntaspclwrzctesx
13 : Drscscmkocobmszrobkvqybsdrw
14 : Cqrbrbljnbnalryqnajupxarcqv
15 : Bpqaqakimamzkqxpmzitowzqbpu
16 : Aopzpzjhlzlyjpwolyhsnvypaot
17 : Znoyoyigkykxiovnkxgrmuxozns
18 : Ymnxnxhfjxjwhnumjwfqltwnymr
19 : Xlmwmwgeiwivgmtlivepksvmxlq
20 : Wklvlvfdhvhuflskhudojrulwkp
21 : Vjkukuecgugtekrjgtcniqtkvjo
22 : Uijtjtdbftfsdjqifsbmhpsjuin
23 : Thisiscaesercipheralgorithm
24 : Sghrhrbzdrdqbhogdqzkfnqhsgl
25 : Rfgqgqaycqcpagnfcpyjempgrfk
```

- Plaintext of "Qefpfpzxbpbozfmeboxidlofqej" is "Thisisceasercipheralgorithm" & key is 23



- Plaintext of "TrvjviTzgyvizjNvrbRcxfizkyd" is "CeaserCipherisWeakAlgorithm" & key is 17



- Plaintext of "LbhNerFzneggbNggnpxPnrfrePvcure" is "YouAreSmarttoAttackCaeserCipher" & key is 13

Practical 4

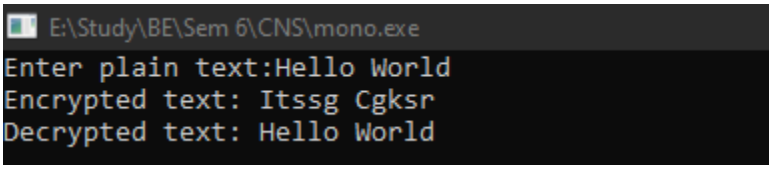
Aim: - Implement a program to perform encryption and decryption using Monoalphabetic cipher algorithm.

- **Code: -**

```
#include<stdio.h>
#include<conio.h>
#define MAX 100

char emap[26] =
{'a','z','e','r','t','y','u','i','o','p','q','s','d','f','g','h','j','k','l','m','w','x','c','v','b','n'};
char plaintext[MAX],ciphertext[MAX],deciphertext[MAX];

void main(){
	int i,ch,index;

	printf("Enter plain text:");
	scanf("%[^\n]",plaintext);

	for(i=0;i<strlen(plaintext);i++){
		ch = plaintext[i];
		if(ch>='A' && ch<='z'){
			if(ch>='a' && ch<='z'){
				index = plaintext[i] - 'a';
				ciphertext[i] = emap[index];
			}
			else if(ch>='A' && ch<='Z'){
				index = plaintext[i] - 'A';
				ciphertext[i] = emap[index] - 32;
			}
		}
		else{
			ciphertext[i] = plaintext[i];
		}
	}
```

```
        printf("Encrypted text: %s",ciphertext);

        for(i=0;i<strlen(ciphertext);i++){
                ch = ciphertext[i];
                if(ch>='A' && ch<='z'){
                        if(ch>='A' && ch<='Z'){
                                index = search(ch+32);
                                deciphertext[i] = index + 'a' - 32;
                        }
                        else{
                                index = search(ch);
                                deciphertext[i] = index + 'a';
                        }
                }
                else{
                        deciphertext[i] = ch;
                }

        }

        printf("\nDecrypted text: %s",deciphertext);
}

int search(int a){
        int i;
        for(i=0;i<strlen(emap);i++){
                if(emap[i]==a){
                        return i;
                }
        }
}
```

- **Output: -**


E:\Study\BE\Sem 6\CNS\mono.exe
Enter plain text:Hello World
Encrypted text: Itssg Cgksr
Decrypted text: Hello World

Practical 5

Aim: - Implement a program to find GCD using Euclidean algorithm & multiplicative inverse using Extended Euclidean algorithm.

- **Code: -**

```
#include<stdio.h>

int mod;

void main(){
	int i,a,b,ans;
	int b3;

	printf("Enter two number for finding GCD: ");
	scanf("%d %d",&a,&b);

	ans = gcd(a,b);
	printf("GCD of (%d,%d) is %d\n\n",a,b,ans);

	printf("Enter m and number:");
	scanf("%d %d",&mod,&b3);

	extended(1,0,mod,0,1,b3);

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

```
void extended(int a1,int a2,int a3,int b1,int b2,int b3){
        int q,t1,t2,t3;

        if(b3==0){
                printf("No inverse");
                return;
        }
        if(b3==1){
                while(b2<0){
                        b2 = b2 + mod;
                }
                printf("Multiplicative Inverse is %d",b2);
                return;
        }
        q = a3/b3;
        t1 = a1-(q*b1);t2 = a2-(q*b2); t3 = a3-(q*b3);
        a1 = b1;a2 = b2;a3 = b3;
        b1 = t1;b2 = t2;b3 = t3;
        extended(a1,a2,a3,b1,b2,b3);
}
```

- **Output: -**

```
Enter two number for finding GCD: 10 25
GCD of (10,25) is 5

Enter m and number:26 7
Multiplicative Inverse is 15
--------------------------------
```

Practical 6

Aim: - Implement a program that returns the value of Euler's totient function.

- **Code: -**

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

int count=0;

void main(){
	int ans,n;

	printf("Enter n:");
	scanf("%d",&n);

	ans = gcd_method(n);
	printf("Using iterative GCD method: %d",ans);

	ans = formula_method(n);
	printf("\nUsing Formula method :%d",ans);
}

int gcd_method(int n){
	int i,count=0;
	for(i=1;i<n;i++){
		if(gcd(i,n)==1){
			count++;
		}
	}
	return count;
}

int formula_method(int n){
	int i,ans=n,a[n];
```

```
        prime_factor(n,a);
        for(i=0;i<count;i++){
                ans = ans * (1.0-(1.0/(float)a[i]));
        }
        return ans;
}

int gcd(int a,int b){
        if(b==0){
                return a;
        }
        return gcd(b,a%b);
}

void prime_factor(int n,int *a){
        int i;
        for(i=2;i<=n/i;i++){
                if(n%i==0){
                        a[count]=i;
                        count++;
                        while(n%i==0){
                                n = n/i;
                        }
                }
        }
        if(n>1){
                a[count]=n;
                count++;
        }
}
```

- **Output: -**

```
Enter n:22
Using iterative GCD method: 10
Using Formula method: 10
--------------------------------
```

Practical 7

Aim: - Implement a program to perform encryption and decryption using Affine cipher algorithm.

- **Code: -**

```
#include<stdio.h>
#define MAX 100

int extended_euclidean(int a1,int a2,int a3,int b1,int b2,int b3){
	int q,t1,t2,t3,mod=26;

	if(b3==0){
		printf("No inverse");
		return;
	}
	if(b3==1){
		while(b2<0){
			b2 = b2 + mod;
		}
		return b2;
	}
	q = a3/b3;
	t1 = a1-(q*b1);t2 = a2-(q*b2); t3 = a3-(q*b3);
	a1 = b1;a2 = b2;a3 = b3;
	b1 = t1;b2 = t2;b3 = t3;
	extended_euclidean(a1,a2,a3,b1,b2,b3);
}
int gcd(int a,int b){
	int temp;
	while(b>0){
		temp = a%b;
		a = b;
		b = temp;
	}
	return a;
```

```
}

void main(){
	int a=0,b=0,i,mul_inv,temp;
	char plaintext[MAX],ciphertext[MAX],deciphertext[MAX];

	printf("Enter plaintext: ");
	scanf("%s",&plaintext);

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

	if(gcd(a,26)!=1){
		return;
	}
	for(i=0;i<strlen(plaintext);i++){
		if(plaintext[i]>='A' && plaintext[i]<='Z'){
			ciphertext[i] = ((a*(plaintext[i] - 'A') + b)%26)+'A';
		}
		else if(plaintext[i]>='a' && plaintext[i]<='z'){
			ciphertext[i] = ((a*(plaintext[i] - 'a') + b)%26)+'a';
		}
		else{
			ciphertext[i] = plaintext[i];
		}
	}

	printf("Encrypted text: %s\n",ciphertext);

	mul_inv = extended_euclidean(1,0,26,0,1,a);

	for(i=0;i<strlen(ciphertext);i++){
		if(ciphertext[i]>='A' && ciphertext[i]<='Z'){
			temp = ((ciphertext[i]-'A')-b)%26;
			if(temp<0){
```

```
                temp += 26;
            }
            deciphertext[i] = ((temp * mul_inv)%26) + 'A';
        }
        else if(plaintext[i]>='a' && plaintext[i]<='z'){
            temp = ((ciphertext[i]-'a')-b)%26;
            if(temp<0){
                temp += 26;
            }
            deciphertext[i] = ((temp * mul_inv)%26) + 'a';
        }
        else{
            deciphertext[i] = ciphertext[i];
        }
    }

    printf("Decrypted text: %s",deciphertext);
}
```

- **Output: -**

```
Enter plaintext: hidden
Enter a: 15
Enter b: 22
Encrypted text: xmppej
Decrypted text: hidden
```

Practical 8

Aim: - Implement a program to perform encryption and decryption using Playfair Cipher.

- **Code: -**

```
#include<stdio.h>
#include<conio.h>
#define MAX 100


void main(){
	char pt[MAX],ct[MAX],dt[MAX],key[MAX],key_table[5][5];
	int i,j;

	printf(" Enter Message: ");
	scanf("%s",&pt);

	printf(" Enter key: ");
	scanf("%s",&key);

	playfair_table(key,key_table);
	printf("\n Playfair Table:\n");
	for(i=0;i<5;i++){
		for(j=0;j<5;j++){
			printf(" %c ",key_table[i][j]);
		}
		printf("\n");
	}
	encrypt(key_table,pt,ct);
	printf("\n Encrypted text: %s",ct);

	decrypt(key_table,ct,dt);
	printf("\n Decrypted text: %s",dt);
}
```

```
void playfair_table(char key[],char key_table[5][5]){
	int taken[26],i,j=0,k,length=strlen(key);

	for(i=0;i<26;i++){
		taken[i]=0;
	}

	for(i=0;i<length;i++){
		if(key[i]!='j'){
			taken[key[i]-'a'] = 2;
		}
	}

	taken['j'-'a'] = 1;

	i=0;j=0;

	for(k=0;k<length;k++){
		if(taken[key[k]-'a']==2){
			taken[key[k]-'a'] = 1;
			key_table[i][j] = key[k];
			j++;
			if(j==5){
				i++;j=0;
			}
		}
	}

	for(k=0;k<26;k++){
		if(taken[k]==0){
			key_table[i][j] = (char)k + 'a';
			j++;
			if(j==5){
				i++;j=0;
			}
		}
	}
```

```
}

void search(char key_table[5][5],char a,char b,int arr[]){
	int i,j;

	if(a=='j'){
		a = 'i';
	}
	if(b=='j'){
		b= 'i';
	}

	for(i=0;i<5;i++){
		for(j=0;j<5;j++){
			if(key_table[i][j]==a){
				arr[0]=i;arr[1]=j;
			}
			if(key_table[i][j]==b){
				arr[2]=i;arr[3]=j;
			}
		}
	}
}

void encrypt(char key_table[5][5],char pt[],char ct[]){
	int i,a[4],length=strlen(pt);

	if(length%2!=0){
		pt[length++] = 'z';
		pt[length] = '\0';
	}

	for(i=0;i<length;i=i+2){
		search(key_table,pt[i],pt[i+1],a);

		if(a[0]==a[2]){
			ct[i] = key_table[a[0]][(a[1]+1)%5];
```

```
                ct[i+1] = key_table[a[2]][(a[3]+1)%5];
            }
            else if(a[1]==a[3]){
                ct[i]= key_table[(a[0]+1)%5][a[1]];
                ct[i+1] = key_table[(a[2]+1)%5][a[3]];
            }
            else{
                ct[i]= key_table[a[0]][a[3]];
                ct[i+1]= key_table[a[2]][a[1]];
            }
        }
}

void decrypt(char key_table[5][5],char ct[],char dt[]){
        int i,a[4],length=strlen(ct);

        for(i=0;i<length;i=i+2){
            search(key_table,ct[i],ct[i+1],a);

            if(a[0]==a[2]){
                dt[i] = key_table[a[0]][(a[1]-1)%5];
                dt[i+1] = key_table[a[2]][(a[3]-1)%5];
            }
            else if(a[1]==a[3]){
                dt[i]= key_table[(a[0]-1)%5][a[1]];
                dt[i+1] = key_table[(a[2]-1)%5][a[3]];
            }
            else{
                dt[i]= key_table[a[0]][a[3]];
                dt[i+1]= key_table[a[2]][a[1]];
            }
        }
}
```

- **Output: -**

```
Enter Message: parth
Enter key: helloworld

Playfair Table:
h  e  l  o  w
r  d  a  b  c
f  g  i  k  m
n  p  q  s  t
u  v  x  y  z

Encrypted text: qdcnwu
Decrypted text: parthz
```

Practical 9

Aim: - Implement a program to perform encryption and decryption using Hill Cipher.

- **Code: -**

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

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();

void 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];

        printf("\nEncrypted string is: ");
        for(i = 0; i < 3; i++)
                printf("%c", (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];

	printf("\nDecrypted string is: ");
	for(i = 0; i < 3; i++)
		printf("%c", (char)(fmod(decrypt[i][0], 26) + 97));

	printf("\n");
}

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

	printf("Enter matrix:\n");

	for(i = 0; i < 3; i++)
		for(j = 0; j < 3; j++) {
			scanf("%f", &a[i][j]);
			c[i][j] = a[i][j];
		}

	printf("\nEnter a 3 letter string: ");
	scanf("%s", 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: -**

```
Enter matrix:
6 24 1 13 16 10 20 17 15

Enter a 3 letter string: std

Encrypted string is: vwa
Decrypted string is: std
```

Practical 10

Aim: - Implement a program to perform encryption and decryption using Rail fence Cipher.

- **Code: -**

```
#include<stdio.h>
#include<conio.h>
#include <stdbool.h>
#define MAX 100

char pt[MAX],ct[MAX],dt[MAX];
int n;

void main(){

	printf("Enter String: ");
	scanf("%s",&pt);

	printf("Enter n: ");
	scanf("%d",&n);

	encrypt();
	printf("Encrypted Text: %s\n",ct);

	decrypt();
	printf("Decrypted Text: %s\n",dt);
}

void encrypt(){
	int j,i;

	j = strlen(pt);
	for(i=0;i<(strlen(pt) % n);i++){
		pt[j++] = 'x';
	}
```

```
        pt[j] = '\0';

        char key[n][j];
        bool flag=true;
        int row=0,col=0;

        for(i=0;i<j;i++){
                key[row][col] = pt[i];

                if(row==0 && i!=0){
                        flag=true;
                        col++;
                        continue;
                }
                else if(row==n-1 && flag==true){
                        flag=false;
                        col++;
                        continue;
                }
                flag?row++:row--;
        }
        int k = floor(strlen(pt) / n);

        int a=0;

        printf("Encryption Table:\n");
        for(i=0;i<n;i++){
                for(j=0;j<k;j++){
                        printf("%c\t",key[i][j]);
                        ct[a] = key[i][j];
                        a++;
                }
                printf("\n");
        }
        ct[a] = '\0';
}
```

```
void decrypt(){
	int row=0,col=0,i,j;
	int k = floor(strlen(ct) / n),a=0;
	char key[n][k];

	for(i=0;i<n;i++){
		for(j=0;j<k;j++){
			key[i][j] = ct[a++];
		}
	}

	j = strlen(ct);
	bool flag=true;
	for(i=0;i<j;i++){
		dt[i] = key[row][col];

		if(row==0 && i!=0){
			flag=true;
			col++;
			continue;
		}
		else if(row==n-1 && flag==true){
			flag=false;
			col++;
			continue;
		}
		flag?row++:row--;
	}
}
```

- **Output: -**

```
Enter String: hello
Enter n: 3
Encryption Table:
h       x
e       o
l       l
Encrypted Text: hxeoll
Decrypted Text: hellox
```

Practical 11

Aim: - Implement a program to perform encryption and decryption using Columnar transposition algorithm.

- **Code: -**

```
#include<stdio.h>
#include<conio.h>
#define MAX 100

char pt[MAX],ct[MAX],dt[MAX];
int keysize,key[MAX];

void main(){
	int i;

	printf("Enter String: ");
	scanf("%s",&pt);

	printf("key size:");
	scanf("%d",&keysize);

	printf("Enter key order: ");
	for(i=0;i<keysize;i++){
		scanf("%d",&key[i]);
	}

	int k = strlen(pt);
	if(strlen(pt)%keysize!=0){
		for(i=0;i<strlen(pt)%keysize;i++){
			pt[k++] = 'x';
		}
	}

	encrypt();

	decrypt();
```

```
}
void encrypt(){
        int i,j,k;
        int m = keysize;
        int n = strlen(pt)/m;
        int ptmatrix[n][m];

        k=0;
        for(i=0;i<n;i++){
                for(j=0;j<m;j++){
                        ptmatrix[i][j] = pt[k];
                        k++;
                }
        }

        int x=1;k=0;
        int l;
        while(x<=m){
                for(l=0;l<m;l++){

                        if(x==key[l]){
                                for(j=0;j<n;j++){
                                        ct[k] = ptmatrix[j][l];
                                        k++;
                                }
                                break;
                        }
                }
                x++;
        }
        ct[k]='\0';

        printf("Encrypted text: %s",ct);
}
```

```
void decrypt(){
	int i,j,k;
	int m = keysize;
	int n = strlen(ct)/m;
	int dtmatrix[n][m];
	int ctmatrix[n][m];

	k=0;
	for(i=0;i<m;i++){
		for(j=0;j<n;j++){
			ctmatrix[j][i]= ct[k];
			k++;
		}
	}

	k=1;
	for(i=0;i<m;i++){
		for(j=0;j<n;j++){
			dtmatrix[j][i] = ctmatrix[j][key[i]-1];
		}
	}

	k=0;
	for(i=0;i<n;i++){
		for(j=0;j<m;j++){
			dt[k] = dtmatrix[i][j];

			k++;
		}
	}
	dt[k] = '\0';

	printf("\nDecrypted Text: %s",dt);
}
```

- **Output: -**

```
Enter String: helloworld
key size:5
Enter key order: 3
5
1
2
4
Encrypted Text: lrllhwodeo
Decrypted Text: helloworld
--------------------------------
```

Practical 12

Aim: - Implement a program to perform encryption and decryption using DES Algorithm.

- **Code: -**

```
#include<stdio.h>

void round(int,int);

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 E[] =
{
    32,  1,  2,  3,  4,  5,
     4,  5,  6,  7,  8,  9,
     8,  9, 10, 11, 12, 13,
    12, 13, 14, 15, 16, 17,
    16, 17, 18, 19, 20, 21,
    20, 21, 22, 23, 24, 25,
    24, 25, 26, 27, 28, 29,
    28, 29, 30, 31, 32,  1
};

int P[] =
{
```

```
    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 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 S1[4][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[4][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[4][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[4][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[4][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[4][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[4][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[4][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
};

int PC1[] =
{
     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 PC2[] =
{
     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 pt[] = {
```

```
        0,1,1,0,0,0,0,1,0,1,1,0,0,0,1,0,0,1,1,0,0,0,1,1,0,1,1,0,0,1,0,0,0,1,1,0,0,1,0,1,0,
1,1,0,0,1,1,0,0,1,1,0,0,1,1,1,0,1,1,0,1,0,0,0
};
int key[] = {
        0,0,1,1,0,0,0,0,0,0,1,1,0,0,0,1,0,0,1,1,0,0,1,0,0,0,1,1,0,0,1,1,0,0,1,1,0,1,0,0,0,
0,1,1,0,1,0,1,0,0,1,1,0,1,1,0,0,0,1,1,0,1,1,1
};
int iptext[64],key56[56],key48[17][48],keycollection[17][56];
int x1[8][6],stext[32];
int temp[64],ct[64];
int left[32],right[32],rightprev[32];

void initialpermutation(){
        int i=0;

        for(i=0;i<64;i++){
                iptext[i] = pt[IP[i]-1];
        }
}
void finalpermutation(){
        int i;

        for(i=0;i<64;i++){
                ct[i] = temp[FP[i]-1];
        }
}
void encryption(){
        int x=1,i,j,k;

        printf("\nPt:\t");
        print(pt,64);
        printf("\nkey:\t");
        print(key,64);

        initialpermutation();
        printf("\nIP:\t");
        print(iptext,64);
```

```
        for(i=0;i<32;i++){
                left[i] = iptext[i];
        }
        for(i=32;i<64;i++){
                right[i-32] = iptext[i];
        }

        for(i=1;i<=16;i++){
                round(i,i);
        }

        for(i=0;i<32;i++){
                temp[i] = right[i];
        }
        for(i=32;i<64;i++){
                temp[i] = left[i-32];
        }

        finalpermutation();
        printf("\n\nCipher Text:\t");
        print(ct,64);

}

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

        for(i=0;i<64;i++){
                pt[i] = ct[i];
        }

        printf("\n\n=============================================
============\nIP:\t");
        print(iptext,64);
        initialpermutation();
```

```
        for(i=0;i<32;i++){
                left[i] = iptext[i];
        }
        for(i=32;i<64;i++){
                right[i-32] = iptext[i];
        }

        printf("\nlt:\t");
        print(left,32);
        printf("\nrt:\t");
        print(right,32);

        k=16;
        for(i=1;i<=16;i++){
                round(i,k);
                k--;
        }

        for(i=0;i<32;i++){
                temp[i] = right[i];
        }
        for(i=32;i<64;i++){
                temp[i] = left[i-32];
        }

        finalpermutation();
        printf("\n\nPlain Text:\t");
        print(ct,64);

}
void print(int arr[],int size){
        int i;
        for (i = 0; i < size; i++){
                if (i % 8 == 0)
                        printf(" ");
                printf("%d", arr[i]);
        }
```

```
}

void round(int x,int key){
        int etext[48],xortext[48],ptext[32];
        int i,j,k;

        printf("\n\n=============================================
=============\nRound %d\n",x);
        printf("\nkey56:\t");
        print(keycollection[key],56);
        printf("\nkey48:\t");
        print(key48[key],48);

        for(i=0;i<48;i++){
                etext[i] = right[E[i]-1];
        }
        printf("\nE:\t");
        print(etext,48);


        for(i=0;i<48;i++){
                xortext[i] = etext[i] ^ key48[key][i];
        }
        printf("\nXOR:\t");
        print(xortext,48);

        k=0;
        for ( i = 0; i < 8; i++){
                for ( j = 0; j < 6; j++){
                        x1[i][j] = xortext[k++];
                }
        }

   sbox();

   printf("\nS-box:\t");
   print(stext,32);
```

```
        for(i=0;i<32;i++){
                ptext[i] = stext[P[i]-1];
        }
        printf("\nP-box:\t");
        print(ptext,32);

        for(i=0;i<32;i++){
                rightprev[i] = right[i];
        }

        for(i=0;i<32;i++){
                right[i] = left[i] ^ ptext[i];
        }
        printf("\nXOR & new rt:\t");
        print(right,32);


   for(i=0;i<32;i++){
        left[i] = rightprev[i];
        }
        printf("\nnew lt:\t");
        print(left,32);
}

void sbox(){
        int value,b[6],r,c,i,j;
   for(i=0;i<8;i++){
        for(j=0;j<6;j++){
                b[j] = x1[i][j];
                }

                r = b[0] * 2 + b[5];
                c = b[1] * 8 + b[2] * 4 + b[3] * 2 + b[4] ;
                switch(i){
                        case 0:
                                value = S1[r][c];
```

```
                        break;
                    case 1:
                        value = S2 [r][c];
                        break;
                    case 2:
                        value = S3 [r][c];
                        break;
                    case 3:
                        value = S4 [r][c];
                        break;
                    case 4:
                        value = S5 [r][c];
                        break;
                    case 5:
                        value = S6 [r][c];
                        break;
                    case 6:
                        value = S7 [r][c];
                        break;
                    case 7:
                        value = S8 [r][c];
                        break;
                }
                tobinary(value);
        }
}

void tobinary(int value){
        int k, j, m;
   static int i;
   if (i % 32 == 0)
      i = 0;
   for (j = 3; j >= 0; j--)
   {
      m = 1 << j;
      k = value & m;
      if (k == 0)
```

```
            stext[3 - j + i] = '0' - 48;
        else
            stext[3 - j + i] = '1' - 48;
    }
    i = i + 4;
}

void key64to56(){
        int i,j=0;
        for(i=0;i<64;i++){
                if((i+1)%8==0){
                        continue;
                }
                key56[j++] = key[i];
        }
}

void keygeneration(){
        int temp[17][2],i,j,k,x;
        int leftkey[17][28],rightkey[17][28],shift;

        key64to56();

        for(i=0;i<28;i++){
                leftkey[0][i] = key56[i];
        }
        for(i=28;i<56;i++){
                rightkey[0][i-28] = key56[i];
        }

        for(x=1;x<17;x++){
                if(x==1 || x==2 || x==9 || x==16){
                        shift=1;
                }
                else{
                        shift=2;
                }
```

```
        for(i=0;i<shift;i++){
                temp[x-1][i] = leftkey[x-1][i];
        }
        for(i=0;i<(28-shift);i++){
                leftkey[x][i] = leftkey[x-1][i+shift];
        }
        k=0;
        for(i=28-shift;i<28;i++){
                leftkey[x][i] = temp[x-1][k++];
        }

        for(i=0;i<shift;i++){
                temp[x-1][i] = rightkey[x-1][i];
        }
        for(i=shift;i<28;i++){
                rightkey[x][i-shift] = rightkey[x-1][i];
        }
        k=0;
        for(i=28-shift;i<28;i++){
                rightkey[x][i] = temp[x-1][k++];
        }
}

for(j=0;j<17;j++){
        for(i=0;i<28;i++){
                keycollection[j][i] = leftkey[j][i];
        }
        for(i=0;i<28;i++){
                keycollection[j][i+28] = rightkey[j][i];
        }
}

for(x=1;x<17;x++){
        for(i=0;i<48;i++){
                key48[x][i] = keycollection[x][PC2[i]-1];
        }
}
```

```
}
void main(){
        keygeneration();
        encryption();
        decryption();
}
```

- **Output: -**

```
Pt:      01100001 01100010 01100011 01100100 01100101 01100110 01100111 01101000
key:     00110000 00110001 00110010 00110011 00110100 00110101 00110110 00110111
IP:      11111111 00000000 01111000 01010101 00000000 11111111 10000000 01100110

============================================================
Round 1

key56:   01100000 11000001 10010011 00100110 10001101 00011011 00110110
key48:   01010010 00011000 00101101 01111111 11111000 00000000
E:       00000000 00010111 11111111 11000000 00000011 00001100
XOR:     01010010 00001111 11010010 10111111 11111011 00001100
S-box:   01100000 11000010 11011101 01111011
P-box:   00111011 01010111 10101111 00001000
XOR & new rt:    11000100 01010111 11010111 01011101
new lt:  00000000 11111111 10000000 01100110

============================================================
Round 2

key56:   11000001 10000011 00100110 01001101 00011010 00110110 01101100
key48:   00001000 10001100 11100001 00011010 10001010 01101111
E:       11100000 10000010 10101111 11101010 11101010 11111011
XOR:     11101000 00001110 01001110 11110000 01100000 10010100
S-box:   10101111 10111010 00001111 10110011
P-box:   01010110 11101010 01101111 01011101
XOR & new rt:    01010110 00010101 11101111 00111011
new lt:  11000100 01010111 11010111 01011101

============================================================
Round 3

key56:   00000110 00001100 10011001 00110100 01101000 11011001 10110011
key48:   11010001 01100000 00011110 11010110 11011101 10010000
E:       10101010 11000000 10101011 11110101 11101001 11110110
XOR:     01111011 10100000 10110101 00100011 00110100 01100110
S-box:   01110011 00000101 01111110 11100001
P-box:   11110000 00110100 11011110 10001011
XOR & new rt:    00110100 01100011 00001001 11010110
```

```
============================================================
Round 14

key56:   00000110 00001100 10011001 00110100 01101000 11011001 10110011
key48:   11010001 01100000 00011110 11010110 11011101 10010000
E:       10101010 11000000 10101011 11110101 11101001 11110110
XOR:     01111011 10100000 10110101 00100011 00110100 01100110
S-box:   01110011 00000101 01111110 11100001
P-box:   11110000 00110100 11011110 10001011
XOR & new rt:    11000100 01010111 11010111 01011101
new lt:  01010110 00010101 11101111 00111011

============================================================
Round 15

key56:   11000001 10000011 00100110 01001101 00011010 00110110 01101100
key48:   00001000 10001100 11100001 00011010 10001010 01101111
E:       11100000 10000010 10101111 11101010 11101010 11111011
XOR:     11101000 00001110 01001110 11110000 01100000 10010100
S-box:   10101111 10111010 00001111 10110011
P-box:   01010110 11101010 01101111 01011101
XOR & new rt:    00000000 11111111 10000000 01100110
new lt:  11000100 01010111 11010111 01011101

============================================================
Round 16

key56:   01100000 11000001 10010011 00100110 10001101 00011011 00110110
key48:   01010010 00011000 00101101 01111111 11111000 00000000
E:       00000000 00010111 11111111 11000000 00000011 00001100
XOR:     01010010 00001111 11010010 10111111 11111011 00001100
S-box:   01100000 11000010 11011101 01111011
P-box:   00111011 01010111 10101111 00001000
XOR & new rt:    11111111 00000000 01111000 01010101
new lt:  00000000 11111111 10000000 01100110

Plain Text:      01100001 01100010 01100011 01100100 01100101 01100110 01100111 01101000
```

Practical 13

Aim: - Implement a program to perform encryption and decryption using RSA Algorithm.

- **Code: -**

```
#include<stdio.h>
#include<math.h>
#include<stdbool.h>
#define MAX 100

bool prime(int);

int p,q,n,t,e,d;
char pt[MAX],ct[MAX],dt[MAX];

void main(){

	printf("Enter message:");
	scanf("%s",&pt);

	do{
		printf("Enter p: ");
		scanf("%d",&p);
	}while(!(prime(p)));

	do{
		printf("Enter q: ");
		scanf("%d",&q);
	}while((!(prime(q))) || q==p);

	n=p*q;
	t=(p-1)*(q-1);

	do{
		printf("Enter e: ");
```

```
            scanf("%d",&e);
        }while(gcd(e,t)!=1 || e>=t);

        d = extended(1,0,t,0,1,e);
        printf("\nD: %d",d);

        encrypt();
        printf("\n\nEncrypted Text: %s",ct);

        decrypt();
        printf("\n\nDecrypted Text: %s",dt);

}
void encrypt(){
        int i,j,len,temp;
        len = strlen(pt);

        for(i=0;i<len;i++){
                temp=1;
                for(j=0;j<e;j++){
                        temp = temp * pt[i];
                        temp = temp % n;
                }
                ct[i] = temp;

        }
        ct[i] = '\0';
}

void decrypt(){
        int i,j,len,temp;
        len = strlen(ct);

        for(i=0;i<len;i++){
                temp=1;
                for(j=0;j<d;j++){
                        temp = temp * ct[i];
```

```
                temp = temp % n;
            }
            dt[i] = temp;
        }
        dt[i] = '\0';
}


bool prime(int a){
        int i;
        for(i=2;i<=sqrt(a);i++){
            if(a%i==0){
                return false;
            }
        }
        return true;
}
int gcd(int a,int b){
        int temp;
        while(b>0){
            temp = a%b;
            a = b;
            b = temp;
        }
        return a;
}
int extended(int a1,int a2,int a3,int b1,int b2,int b3){
        int q,t1,t2,t3;

        if(b3==0){
            return;
        }
        if(b3==1){
            while(b2<0){
                b2 = b2 + t;
            }
            return b2;
```

```
        }
        q = a3/b3;
        t1 = a1-(q*b1);t2 = a2-(q*b2); t3 = a3-(q*b3);
        a1 = b1;a2 = b2;a3 = b3;
        b1 = t1;b2 = t2;b3 = t3;
        extended(a1,a2,a3,b1,b2,b3);
}
```

- **Output: -**

```
Enter message:parth
Enter p: 7
Enter q: 17
Enter e: 5
D: 77
Encrypted Text: [¶XrS
Decrypted Text: parth
```

Practical 14

Aim: - Implement a program to generate key using Diffie-Hellman key exchange algorithm and using that key perform encryption and decryption using Caesar cipher algorithm.

- **Code: -**

```
#include<stdio.h>
#define MAX 100

char pt[MAX],ct[MAX],dt[MAX];
int p,g,pua,pra,pub,prb,ka,kb;


void main(){
        int i,j;
        char ch,temp;

        printf("Enter P and G: ");
        scanf("%d %d",&p,&g);

        printf("Enter private key of a: ");
        scanf("%d",&pra);

        printf("Enter private key of b: ");
        scanf("%d",&prb);

        pua = power(g,pra);
        pua = pua % p;

        pub = power(g,prb);
        pub = pub % p;

        ka = power(pub,pra);
        ka = ka % p;
```

```
kb = power(pua,prb);
kb = kb % p;

printf("Secret key: %d(a side) %d(b side)",ka,kb);

printf("\nEnter Message: ");
scanf("%c",&temp);
scanf("%[^\n]",&pt);

if(ka>26){
	ka = ka%26;
}
for(i=0;i<strlen(pt);i++){
	ch = pt[i];
	if(ch>='a' && ch<='z'){
		ch = ch + ka;
		if(ch>'z'){
			ch = ch - 'z' + 'a' - 1;
		}
		ct[i] = ch;
	}
	else if(ch>='A' && ch<='Z'){
		ch = ch + ka;
		if(ch>'Z'){
			ch = ch - 'Z' + 'A' - 1;
		}
		ct[i] = ch;
	}
	else{
		ct[i] = ch;
	}
}
ct[i]='\0';

printf("Encrypted text: %s",ct);

if(kb>26){
```

```
            kb = kb%26;
        }
        for(i=0;i<strlen(ct);i++){
            ch = ct[i];

            if(ch>='a' && ch<='z'){
                ch = ch - kb;
                if(ch<'a'){
                    ch = ch + 'z' - 'a' + 1;
                }
                dt[i] = ch;
            }
            else if(ch>='A' && ch<='Z'){
                ch = ch - kb;
                if(ch<'A'){
                    ch = ch + 'Z' - 'A' + 1;
                }
                dt[i] = ch;
            }
            else{
                dt[i] = ch;
            }
        }

        printf("\nDecrypted text: %s",dt);
}
int power(int a,int b){
        int i,temp=1;
        for(i=0;i<b;i++){
            temp = temp * a;
        }
        return temp;
}
```

- **Output: -**

```
Enter P and G: 23 9
Enter private key of a: 4
Enter private key of b: 3
Secret key: 9(a side) 9(b side)
Enter Message: parth gabani
Encrypted text: yjacq pjkjwr
Decrypted text: parth gabani
```