CRPTOGRAPHY AND NETWORK SECURITY LAB

Index

Sr. No.	Practical
1.	Program to implement Caesar Cipher.
2.	Program to implement Mono alphabetic Cipher.
3.	Networking: a. How to find the OS of the target machine: i. Daemon grabbing ii. Active fingerprinting iii. ICMP messages iv. Passive fingerprinting b. To get list of services running on various open ports. c. How to take the information of system where the IP address of target, subnet mask i. Traceroute ii. ICMP
4.	Program to implement Transposition Cipher.
5.	Program to implement Hill Cipher.
6.	Program to find the GCD of two polynomials using Euclidean.
7.	Program to find the multiplicative inverse of a number.
8.	Program to implement Play fair Cipher.
9.	Program to implement Rail fence Cipher.
10.	Program to implement simplified AES.
11.	Program to check primality using Miller – Rabin theorem.

12.	Program to solve the equations using Chinese Remainder theorem.
13.	Program to encrypt and decrypt the text using DES.
14.	Program to implement fast exponentiation.
15.	Program to implement RSA algorithm.
16.	Program to implement text cover.
17.	Program to implement random number generator.
18.	Program to implement Discrete algorithm.
19.	Program to implement DSA algorithm.
20.	Program to implement Elgamal DSA.
21.	Program to implement RSA DSA.
22.	Program to implement Diffie Hellman algorithm.

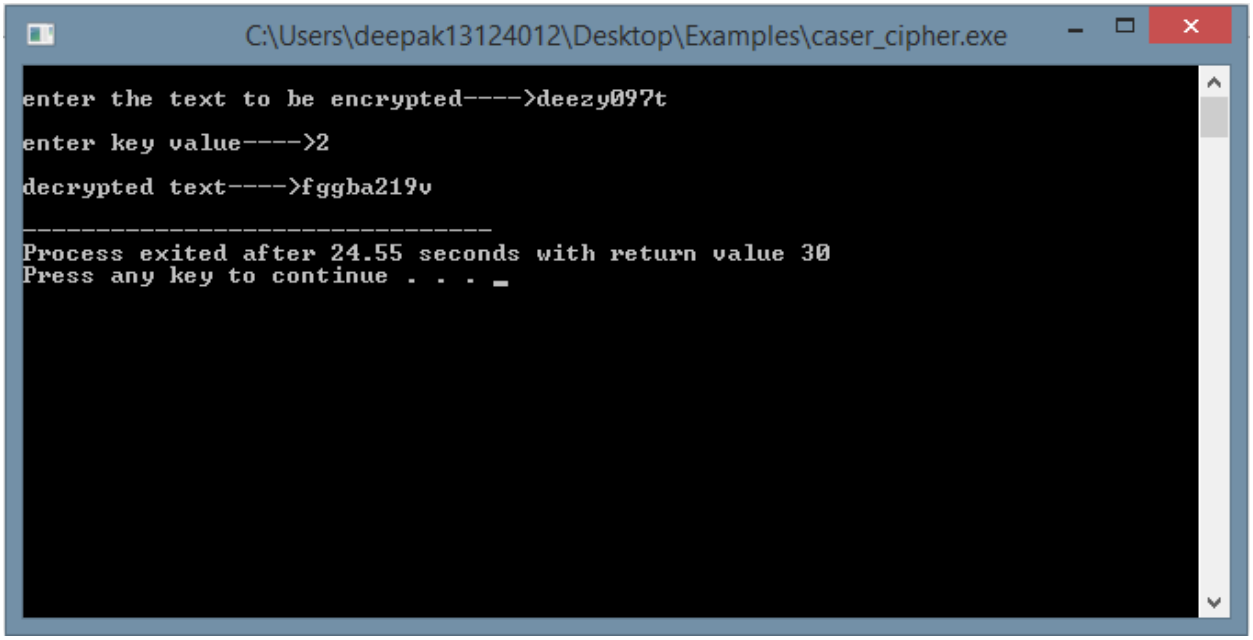
1. Program to implement Caesar Cipher.

Program:

```
#include<stdio.h>
#include<string.h>
void main()
{
		FILE *fp;
		printf("\nenter the text to be encrypted---->");
		fp=fopen("encrypt.txt","w");
		char ch;
		char str[100];
		while((ch=getc(stdin))!='\n')
		{
			fputc(ch,fp);
		}

		fclose(fp);
		int k;
		int i=0;
		printf("\nenter key value---->");
		scanf("%d",&k);
		FILE *f=fopen("encrypt.txt","r");
		FILE *d=fopen("decrypt.txt","w");
		while((ch=fgetc(f))!=EOF)
		{
			int val=(int)ch;
			if(val>=65&&val<=90)
			{
				val=val+k;
				if(val<65||val>90)
					val=(val%91)+65;
			}
			else
			if(val>=97&&val<=122)
			{
				val=val+k;if(val<97||val>122)
					val=(val%123)+97;
			}
			else
			if(val>=48&&val<=57)
			{
				val=val+k;
				if(val<48||val>57)
```

```
                                        val=(val%58)+48;
                        }
                        ch=(char)val;
                        str[i++]=ch;
                }
                str[i]='\0';
                printf("\ndecrypted text---->%s\n",str);
}
```

Output:



2. Program to implement Mono alphabetic Cipher.

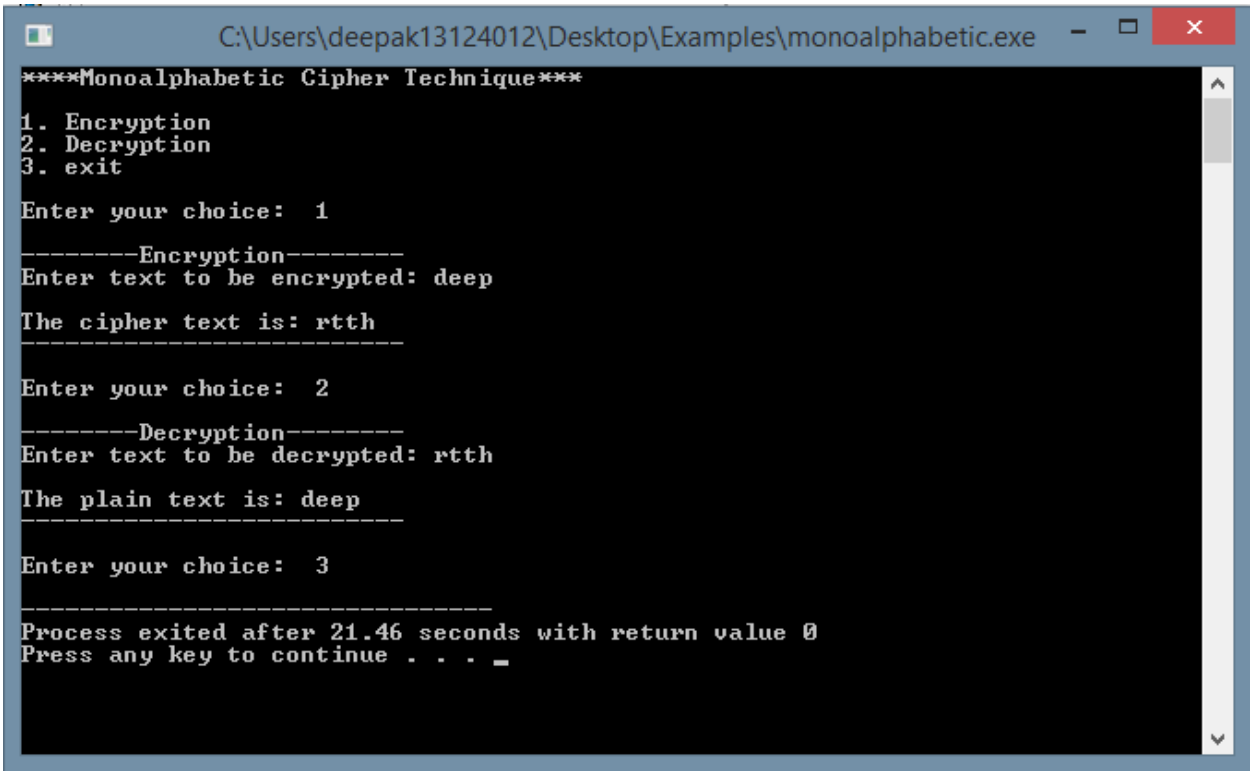
Program:

```
#include<iostream>
#include<conio.h>
#include<stdio.h>
#include<string.h>
using namespace std;
void encryption();
void decryption();
char pt[50],ct[50],ch;
char alpha[26]={'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z'};
char sub[26]=  {'q','w','e','r','t','y','u','i','o','p','a','s','d','f','g','h','j','k','l','z','x','c','v','b','n','m'};
int i,j;
int main()
{
                int choice;
                cout<<"****Monoalphabetic Cipher Technique***";
                cout<<"\n\n1. Encryption\n2. Decryption\n3. exit";
                while(1)
                {
                        cout<<"\n\nEnter your choice:  ";
                        cin>>choice;
                        switch(choice)
                        {
                                case 1: //Encryption
                                        encryption();
                                        break;
```

```
                        case 2: //Decryption
                                decryption();
                                break;
                        case 3: return 0;
                                break;
                        default: //Wrong Input
                        cout<<"\nIncorrect choice. Try again!!";
                        break;
                }
        }
        getch();
}
void encryption()
{
        cout<<"\n--------Encryption--------";
        cout<<"\nEnter text to be encrypted: ";
        scanf("%s",pt);
        for(i=0;i<strlen(pt);i++)
        {
                ch = pt[i];
                for(j=0;j<26;j++)
                {
                        if(alpha[j]==ch)
                        {
                                ct[i]=sub[j];
                                // break;
                        }
                }
        }
        ct[i]='\n';
```

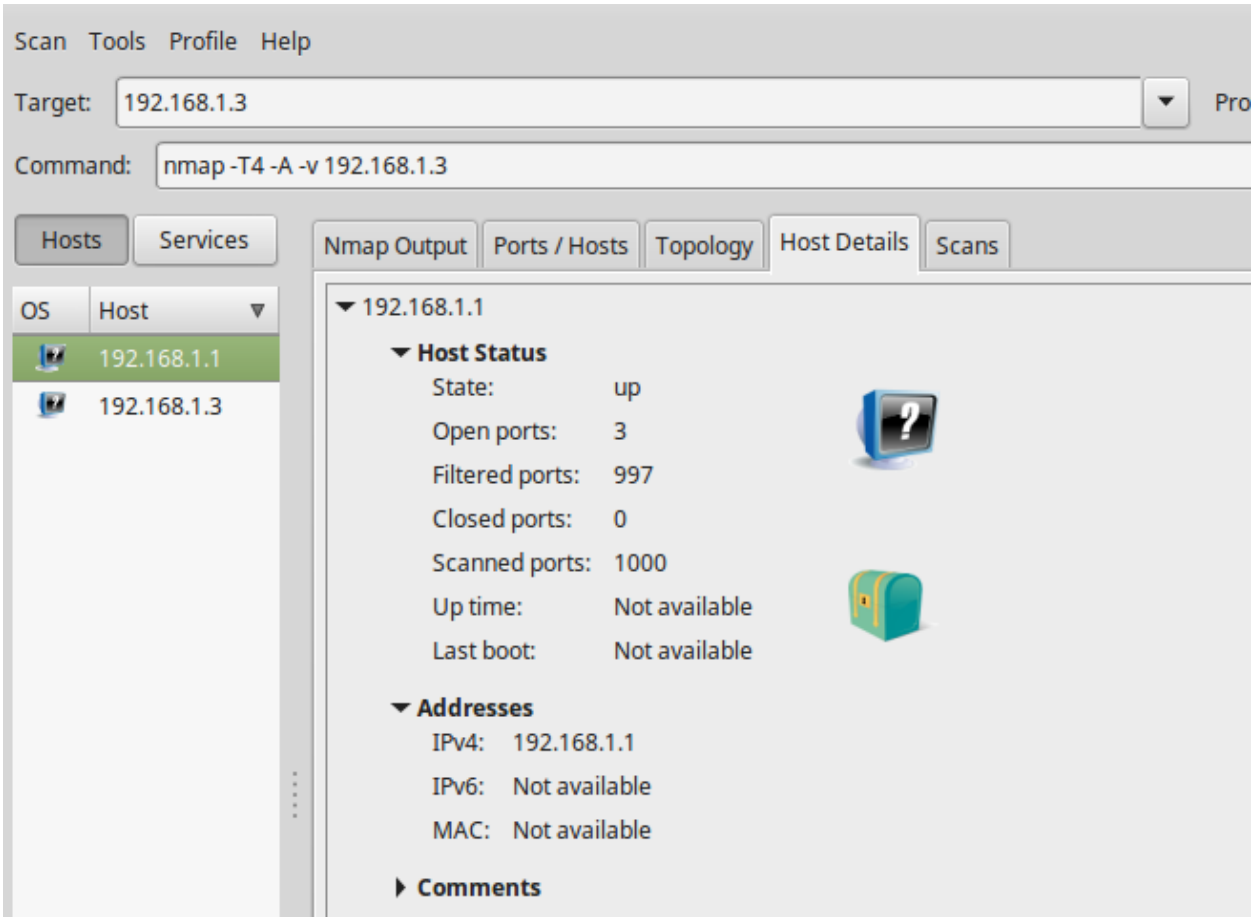
```
		cout<<"\nThe cipher text is: ";
		for(i=0;i<strlen(pt);i++)
			cout<<ct[i];
		cout<<"\n--------------------------";
}
void decryption()
{
		cout<<"\n--------Decryption--------";
		cout<<"\nEnter text to be decrypted: ";
		scanf("%s",ct);
		for(i=0;i<strlen(ct);i++)
		{
			ch=ct[i];
			for(j=0;j<26;j++)
			{
				if(sub[j]==ch)
				{
					pt[i]=alpha[j];
					break;
				}
			}
		}
		pt[i]=' ';
		cout<<"\nThe plain text is: ";
		for(i=0;i<strlen(ct);i++)
			cout<<pt[i];
		cout<<"\n--------------------------";
}
```

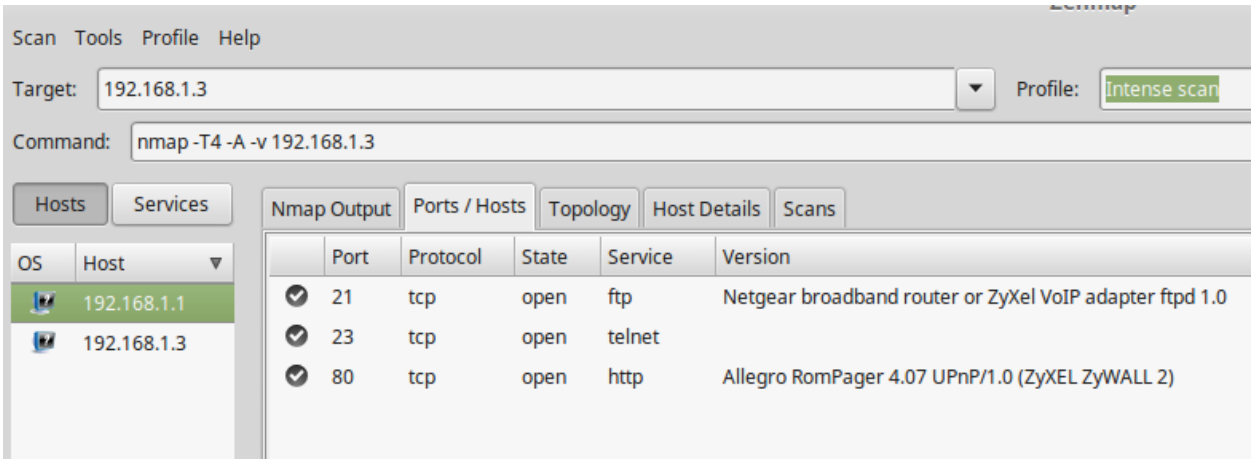
Output:

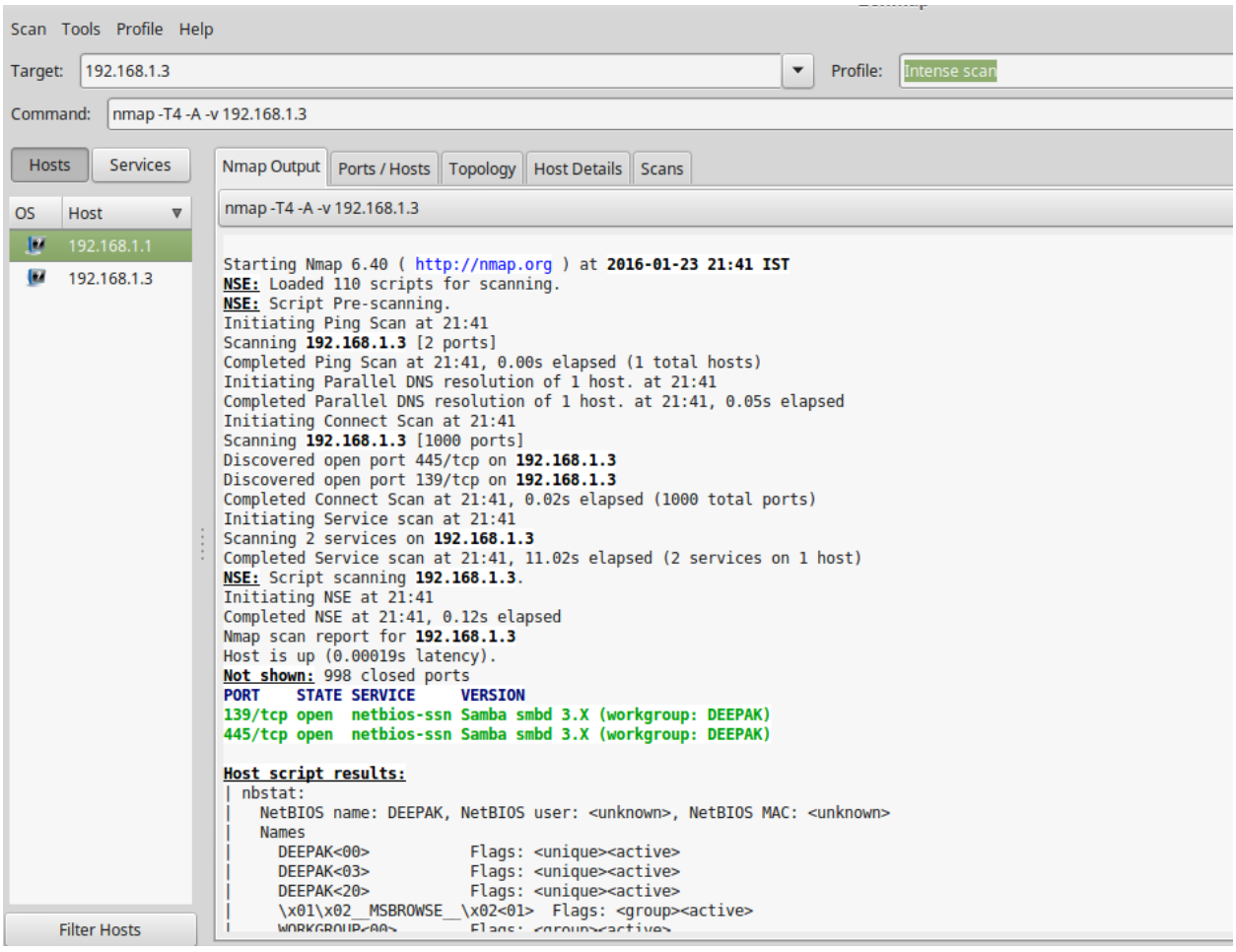

C:\Users\deepak13124012\Desktop\Examples\monoalphabetic.exe
****Monoalphabetic Cipher Technique***
1. Encryption
2. Decryption
3. exit
Enter your choice: 1
--------Encryption--------
Enter text to be encrypted: deep
The cipher text is: rtth
Enter your choice: 2
--------Decryption--------
Enter text to be decrypted: rtth
The plain text is: deep
Enter your choice: 3
Process exited after 21.46 seconds with return value 0
Press any key to continue . . .

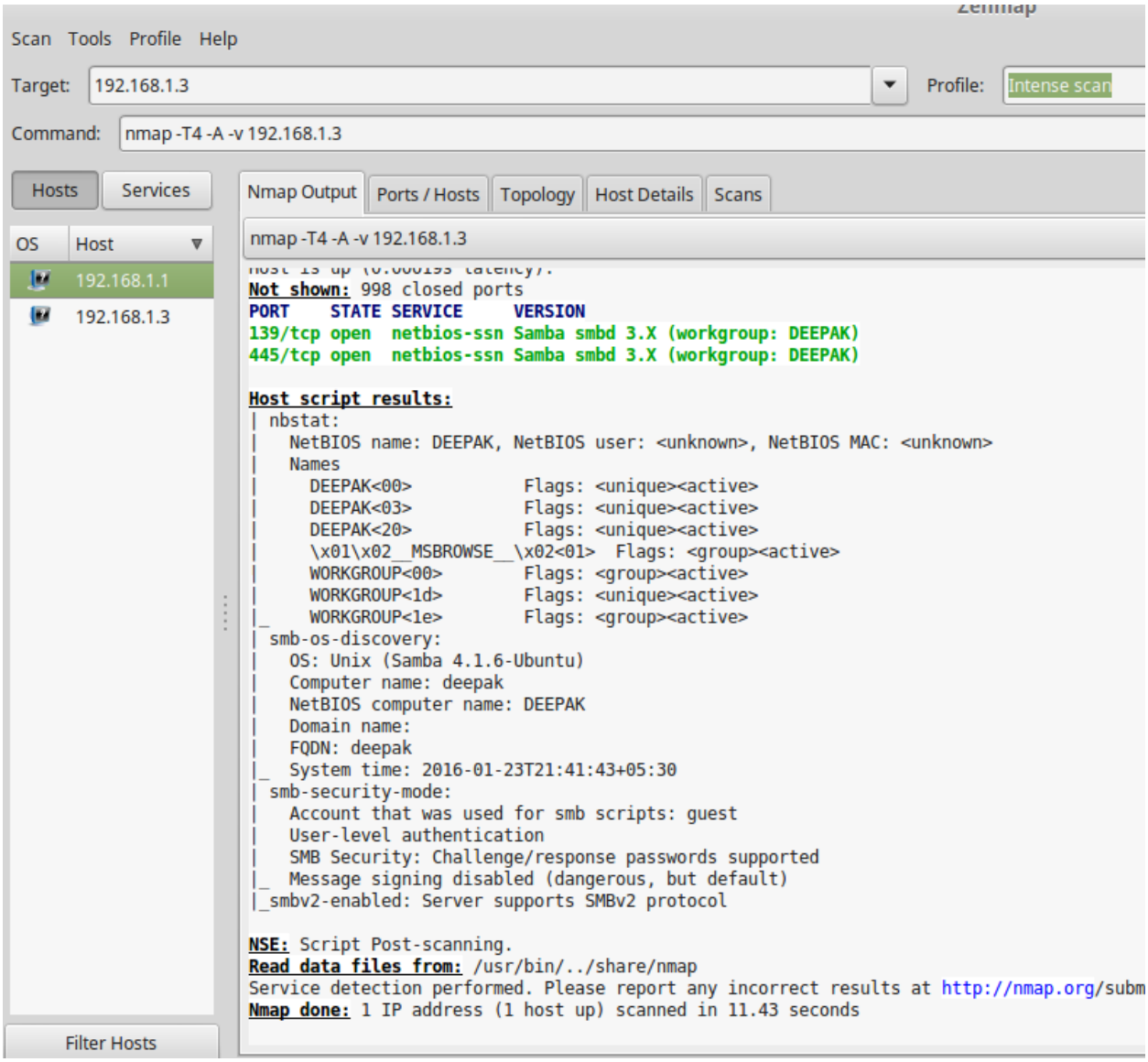
3. **Networking:**
   a. **How to find the OS of the target machine:**
      v. **Daemon grabbing**
      vi. **Active fingerprinting**
      vii. **ICMP messages**
      viii. **Passive fingerprinting**
   b. **To get list of services running on various open ports.**
   c. **How to take the information of system where the IP address of target, subnet mask**
      iii. **Traceroute**
      iv. **ICMP**

Output:




Scan Tools Profile Help
Target: 192.168.1.3
Profile: Intense scan
Command: nmap -T4 -A -v 192.168.1.3
Hosts
Services
Nmap Output
Ports / Hosts
Topology
Host Details
Scans
OS
Host
192.168.1.1
192.168.1.3
Port
Protocol
State
Service
Version
21 tcp open ftp Netgear broadband router or ZyXel VoIP adapter ftpd 1.0
23 tcp open telnet
80 tcp open http Allegro RomPager 4.07 UPnP/1.0 (ZyXEL ZyWALL 2)


Scan Tools Profile Help
Target: 192.168.1.3
Profile: Intense scan
Command: nmap -T4 -A -v 192.168.1.3
Hosts
Services
Nmap Output
Ports / Hosts
Topology
Host Details
Scans
OS
Host
192.168.1.1
192.168.1.3
Filter Hosts
nmap -T4 -A -v 192.168.1.3
Starting Nmap 6.40 (http://nmap.org) at 2016-01-23 21:41 IST
NSE: Loaded 110 scripts for scanning.
NSE: Script Pre-scanning.
Initiating Ping Scan at 21:41
Scanning 192.168.1.3 [2 ports]
Completed Ping Scan at 21:41, 0.00s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 21:41
Completed Parallel DNS resolution of 1 host. at 21:41, 0.05s elapsed
Initiating Connect Scan at 21:41
Scanning 192.168.1.3 [1000 ports]
Discovered open port 445/tcp on 192.168.1.3
Discovered open port 139/tcp on 192.168.1.3
Completed Connect Scan at 21:41, 0.02s elapsed (1000 total ports)
Initiating Service scan at 21:41
Scanning 2 services on 192.168.1.3
Completed Service scan at 21:41, 11.02s elapsed (2 services on 1 host)
NSE: Script scanning 192.168.1.3.
Initiating NSE at 21:41
Completed NSE at 21:41, 0.12s elapsed
Nmap scan report for 192.168.1.3
Host is up (0.00019s latency).
Not shown: 998 closed ports
PORT STATE SERVICE VERSION
139/tcp open netbios-ssn Samba smbd 3.X (workgroup: DEEPAK)
445/tcp open netbios-ssn Samba smbd 3.X (workgroup: DEEPAK)
Host script results:
| nbstat:
| NetBIOS name: DEEPAK, NetBIOS user: <unknown>, NetBIOS MAC: <unknown>
| Names
| DEEPAK<00> Flags: <unique><active>
| DEEPAK<03> Flags: <unique><active>
| DEEPAK<20> Flags: <unique><active>
| \x01\x02__MSBROWSE__\x02<01> Flags: <group><active>




```
deepak@deepak / $ nmap -sT -p- -Pn 192.168.1.3

Starting Nmap 6.40 ( http://nmap.org ) at 2016-01-23 21:33 IST
Nmap scan report for 192.168.1.3
Host is up (0.000076s latency).
Not shown: 65533 closed ports
PORT    STATE SERVICE
139/tcp open  netbios-ssn
445/tcp open  microsoft-ds

Nmap done: 1 IP address (1 host up) scanned in 0.59 seconds
deepak@deepak / $
```

```
deepak@deepak / $ xprobe2 google.com

Xprobe2 v.0.3 Copyright (c) 2002-2005 fyodor@o0o.nu, ofir@sys-security.com, mede
r@o0o.nu

[+] Target is google.com
[+] Loading modules.
[+] Following modules are loaded:
[x] [1] ping:icmp_ping  -  ICMP echo discovery module
[x] [2] ping:tcp_ping  -  TCP-based ping discovery module
[x] [3] ping:udp_ping  -  UDP-based ping discovery module
[x] [4] infogather:ttl_calc  -  TCP and UDP based TTL distance calculation
[x] [5] infogather:portscan  -  TCP and UDP PortScanner
```

```
deepak@deepak / $ traceroute 127.0.0.1
traceroute to 127.0.0.1 (127.0.0.1), 30 hops max, 60 byte packets
 1  localhost (127.0.0.1)  0.036 ms  0.007 ms  0.007 ms
deepak@deepak / $ traceroute google.com
traceroute to google.com (216.58.220.14), 30 hops max, 60 byte packets
 1  192.168.1.1 (192.168.1.1)  0.622 ms  1.117 ms  6.221 ms
 2  117.203.128.1 (117.203.128.1)  21.524 ms  25.172 ms  26.619 ms
 3  218.248.162.150 (218.248.162.150)  28.281 ms  32.523 ms  34.925 ms
 4  218.248.235.129 (218.248.235.129)  54.930 ms *  52.523 ms
 5  218.248.235.130 (218.248.235.130)  56.763 ms * *
 6  72.14.218.242 (72.14.218.242)  82.203 ms 72.14.218.234 (72.14.218.234)  50.4
15 ms  50.946 ms
 7  66.249.95.106 (66.249.95.106)  63.934 ms  60.739 ms  64.330 ms
 8  209.85.252.143 (209.85.252.143)  67.841 ms  69.067 ms  73.511 ms
 9  74.125.37.235 (74.125.37.235)  73.989 ms  76.275 ms  78.900 ms
10  bom05s05-in-f14.1e100.net (216.58.220.14)  87.285 ms  87.994 ms  55.030 ms
deepak@deepak / $
```

```
C:\Users\deepak13124012>ping  10.10.128.44

Pinging 10.10.128.44 with 32 bytes of data:
Reply from 10.10.128.80: Destination host unreachable.
Reply from 10.10.128.80: Destination host unreachable.
Reply from 10.10.128.80: Destination host unreachable.

Ping statistics for 10.10.128.44:
    Packets: Sent = 3, Received = 3, Lost = 0 (0% loss),
Control-C
^C
C:\Users\deepak13124012>ping  10.10.128.80

Pinging 10.10.128.80 with 32 bytes of data:
Reply from 10.10.128.80: bytes=32 time<1ms TTL=128
Reply from 10.10.128.80: bytes=32 time<1ms TTL=128
Reply from 10.10.128.80: bytes=32 time<1ms TTL=128
Reply from 10.10.128.80: bytes=32 time<1ms TTL=128

Ping statistics for 10.10.128.80:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\Users\deepak13124012>
```

4. Program to implement Transposition Cipher.

Program:

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
void cipher(int i,int c);
int findMin();
void makeArray(int,int);
char arr[22][22],darr[22][22],emessage[111],retmessage[111],key[55];
char temp[55],temp2[55];
int k=0;
int main() {
        char *message,*dmessage;
        int i,j,klen,emlen,flag=0;
        int r,c,index,min,rows;

        FILE *fp;
        printf("\nenter the text to be encrypted---->");
        fp=fopen("encrypt.txt","w");
        char ch;
        char str[100];
        while((ch=getc(stdin))!='\n')
        {
                fputc(ch,fp);
        }

        fclose(fp);
        printf("Enetr the key\n");
        fflush(stdin);
        scanf("%s",key);
        FILE *f=fopen("encrypt.txt","r");
        i=0;
        while((ch=fgetc(f))!=EOF)
        {
                emessage[i]=ch;
                i++;
        }
        message=&emessage[0];
        strcpy(temp,key);
        klen=strlen(key);
        k=0;
        for (i=0; ;i++) {
                if(flag==1)
```

```
            break;
        for (j=0;key[j]!='\0';j++) {
            if(message[k]=='\0') {
                flag=1;
                arr[i][j]='-';
            } else {
                arr[i][j]=message[k++];
            }
        }
}
r=i;
c=j;
for (i=0;i<r;i++) {
        for (j=0;j<c;j++) {
            printf("%c ",arr[i][j]);
        }
        printf("\n");
}
k=0;
for (i=0;i<klen;i++) {
        index=findMin();
        cipher(index,r);
}
emessage[k]='\0';
printf("\nEncrypted message is\n");
for (i=0;emessage[i]!='\0';i++)
   printf("%c",emessage[i]);
printf("\n\n");
//deciphering
emlen=strlen(emessage);
//emlen is length of encrypted message
strcpy(temp,key);
rows=emlen/klen;
//rows is no of row of the array to made from ciphered message
rows;
j=0;
for (i=0,k=1;emessage[i]!='\0';i++,k++) {
        //printf("\nEmlen=%d",emlen);
        temp2[j++]=emessage[i];
        if((k%rows)==0) {
            temp2[j]='\0';
            index=findMin();
            makeArray(index,rows);
            j=0;
        }
}
```

```
	printf("\nArray Retrieved is\n");
	k=0;
	for (i=0;i<r;i++) {
		for (j=0;j<c;j++) {
			printf("%c ",darr[i][j]);
			//retrieving message
			retmessage[k++]=darr[i][j];
		}
		printf("\n");
	}
	retmessage[k]='\0';
	printf("\nMessage retrieved is\n");
	for (i=0;retmessage[i]!='\0';i++)
	    printf("%c",retmessage[i]);
	//getch();
	return(0);
}
void cipher(int i,int r) {
	int j;
	for (j=0;j<r;j++) { {
			emessage[k++]=arr[j][i];
		}
	}
	// emessage[k]='\0';
}
void makeArray(int col,int row) {
	int i,j;
	for (i=0;i<row;i++) {
		darr[i][col]=temp2[i];
	}
}
int findMin() {
	int i,j,min,index;
	min=temp[0];
	index=0;
	for (j=0;temp[j]!='\0';j++) {
		if(temp[j]<min) {
			min=temp[j];
			index=j;
		}
	}
	temp[index]=123;
	return(index);
}
```

Output:

```
deepak@deepak ~/cryptography $ gcc transposition.c
deepak@deepak ~/cryptography $ ./a.out

enter the text to be encrypted---->how are you
Enetr the key
hello
h o w   a
r e   y o
u - - - -

Encrypted message is
oe-hruw - y-ao-


Array Retrieved is
h o w   a
r e   y o
u - - - -

Message retrieved is
how are you----deepak@deepak ~/cryptography $
```

5. Program to implement Hill Cipher.

Program:

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
int **matrixMultiply(int**a,int r1,int c1,int **b,int r2,int c2)
{
 int **resultMatrix;
 int i,j,k,r,c;
 r=r1;c=c2;
 resultMatrix=(int**)malloc(sizeof(int*)*r);
 for(i=0;i<r;i++)
  resultMatrix[i]=(int*)malloc(sizeof(int)*c);
 for(i=0;i<r;i++)
 {
  for(j=0;j<c;j++)
  {
   resultMatrix[i][j]=0;
   for(k=0;k<c1;k++)
    resultMatrix[i][j]+=a[i][k]*b[k][j];
  }
 }
 return resultMatrix;
}
void printMatrix(int**matrix,int r,int c)
{
 int i,j;
 for(i=0;i<r;i++)
 {
  for(j=0;j<c;j++)
   printf("%d ",matrix[i][j]);
  printf("\n");
 }
}
int plainTextToCipherText(char plainText[],int**matrix)
{
 int len,**plainTextMatrix,**resultMatrix,i,j;
 // The matrix will be of dimensions strlen(plainText) by strlen(plainText)
 char *cipherText;
 len=strlen(plainText);
 cipherText=(char*)malloc(sizeof(char)*1000);
```

```
 // plainTextMatrix should be of dimension strlen(plainText) by 1
 // allcating memory to plainTextMatrix
 plainTextMatrix=(int**)malloc(sizeof(int*)*len);
 for(i=0;i<len;i++)
  plainTextMatrix[i]=(int*)malloc(sizeof(int)*1);

 // populating the plainTextMatrix
 for(i=0;i<len;i++)
  for(j=0;j<1;j++)
   plainTextMatrix[i][j]=plainText[i]-'a';

 resultMatrix=matrixMultiply(matrix,len,len,plainTextMatrix,len,1);

 // taking mod 26 of each element of the result matrix
 for(i=0;i<len;i++)
  for(j=0;j<1;j++)
   resultMatrix[i][j]%=26;

 // Printing the cipher text
 printf("The cipher text is as follows : ");
 for(i=0;i<len;i++)
  for(j=0;j<1;j++)
   printf("%c",resultMatrix[i][j]+'a');
 printf("\n");
 //printMatrix(resultMatrix,len,1);
}
int main()
{
 int len,i,j,**matrix;
 char plainText[1000];
 printf("Enter the word to be encrypted : ");
 scanf(" %s",plainText);
 len=strlen(plainText);

 // allocating memory to matrix
 matrix=(int**)malloc(sizeof(int*)*len);
 for(i=0;i<len;i++)
  matrix[i]=(int*)malloc(sizeof(int)*len);

 printf("Enter the matrix of %d by %d to be used in encryption process : \n",len,len);
 for(i=0;i<len;i++)
  for(j=0;j<len;j++)
   scanf("%d",&matrix[i][j]);

 plainTextToCipherText(plainText,matrix);
 return 0;}
```

Output:

```
deepak@deepak ~/cryptography $ ./a.out
Enter the word to be encrypted : hel
Enter the matrix of 3 by 3 to be used in encryption process :
1       2       3       4       5       6       7
8       9
The cipher text is as follows : wky
deepak@deepak ~/cryptography $
```

6. Program to find the GCD of two polynomials using Euclidean.

Program:

```
#include<stdio.h>
int power(int x,int y){
        int i;
        int r=1;
        for(i=0;i<y;i++){
                r=r*x;
        }
        return r;
}
int sub(int a,int b,int m){
        int f=0,s,t,r,k,i;
        for(i=1;i<=m;i++){
                k=10;
                s=a%k;
                t=b%k;
                a=a/k;
                b=b/k;
                if(s==t)
                        r=0;
                if(s!=0&&t==0)
                        r=1;
                if(t!=0&&s==0)
                        r=1;
                s=i;
                k=power(10,s-1);
                f=f+r*k;
```

```
        }
        return f;
}
int bits(int x){
        int s=0;
        while(x){
                x=x/10;
                s++;
        }
        return s;
}
int gcd(int a,int b,int m){
        int diff,r,temp,final;
        while(bits(a)>=bits(b)){
                diff=bits(a)-bits(b);
                if(diff>=0)
                        temp=b*power(10,diff);
                r=sub(a,temp,m);
                final=a;
                a=r;
        }
        if(a==0)        return final;
        if(bits(a)<bits(b))    gcd(b,a,m);
}
int makenumber(int a[], int m){
        int r=0,i;
        int temp=m;
        for(i=0;i<m;i++){
                temp--;
                r=r+a[i]*power(10,temp);
```

```
        }
        return r;
}
void main(){
        int i,j,m;
        printf("\nenter the highest power of first equation");
        scanf("%d",&m);
        int arr1[m],arr2[m];
        printf("\nenter the coefficients of first equation in decreasing order of power");
        printf("starting from coefficient of highest degree %d\n",m);
        for(i=0;i<=m;i++){
                scanf("%d",&arr1[i]);
        }
        printf("\nenter the coefficients of second equation in decreasing order of power");
        printf("starting from coefficient of highest degree %d\n",m);
        for(i=0;i<=m;i++){
                scanf("%d",&arr2[i]);
        }
        printf("\nfirst equation is:   ");
        for(i=m,j=0;i>=0,j<=m;i--,j++){
                if(j!=m)                printf("%dx^%d + ",arr1[j],i);
                else  printf("%d",arr1[j]);
        }
        printf("\nsecond equation is:   ");
        for(i=m,j=0;i>=0,j<=m;i--,j++){
                if(j!=m)                printf("%dx^%d + ",arr2[j],i);
                else printf("%d",arr2[j]);
        }
        int a=makenumber(arr1,m+1);
        int b=makenumber(arr2,m+1);
```

```
        m++;
        int r= gcd(a,b,m);
        printf("\nGCD:   ");
        for(i=bits(r)-1;i>=0;i--){
                int t=r/power(10,i);
                if(i!=0) printf("%dx^%d +",t,i);
                else     printf("%d\n",t );
                r=r%power(10,i);
        }
}
```

Output:

```
deepak@deepak ~/cryptography $ gcc gcd.c
deepak@deepak ~/cryptography $ ./a.out

enter the highest power of first equation6

enter the coefficients of first equation in decreasing order of powerstarting fr
om coefficient of highest degree 6
1 1 1 1 1 1 1

enter the coefficients of second equation in decreasing order of powerstarting f
rom coefficient of highest degree 6
0 0 1 0 1 1 1

first equation is:   1x^6 + 1x^5 + 1x^4 + 1x^3 + 1x^2 + 1x^1 + 1
second equation is:   0x^6 + 0x^5 + 1x^4 + 0x^3 + 1x^2 + 1x^1 + 1
GCD:   1x^3 +1x^2 +0x^1 +1
deepak@deepak ~/cryptography $
```

7. Program to find the multiplicative inverse of a number.

Program:

```
#include <stdio.h>
int modInverse(int a, int m){
	int m0 = m, t, q, x0 = 0, x1 = 1;
	if (m == 1)	return 0;
	while (a > 1){
		q = a / m;
		t = m;
		m = a % m, a = t;
		t = x0;
		x0 = x1 - q * x0;
			x1 = t;
	}
	if (x1 < 0)	x1 += m0;
	return x1;
}
int main(){
	int a,m;
	printf("\nenter the value of 'a' and 'm' where 'a' under modulo 'm'");
	scanf("%d%d",&a,&m);
	printf("Modular multiplicative inverse is %d\n",
		modInverse(a, m));
	return 0;
}
```

Output:

```
deepak@deepak ~/cryptography $ gcc exted_euclidean.c
deepak@deepak ~/cryptography $ ./a.out

enter the value of 'a' and 'm' where 'a' under modulo 'm'3
11
Modular multiplicative inverse is 4
deepak@deepak ~/cryptography $
```

8. Program to implement Play fair Cipher.

Program:

```
#include <bits/stdc++.h>
using namespace std;
static int substitution_index;
char *mat[8];
void build_matrix(string &s1,char ** mat){
	for(int i=0 ; i<8; i++)
		mat[i]= new char[8];
	for(int i=0; i<8; i++)
		for(int j=0; j<8; j++)
			mat[i][j]='!';
	set<char> unique;
	for(int i=0; i<s1.size(); i++)
		unique.insert(s1[i]);
	set<char>::iterator sit=unique.begin();
	int last_row,last_col;
	bool over=false;
	for(int i=0; i<8; i++){
		for(int j=0; j<8; j++)
			if (sit!=unique.end()){
				mat[i][j]=*sit;
				sit++;
			}
			else{
				last_col=j;
				last_row=i;
				over=true;
```

```
                        break;
                }
        if (over)break;
}
vector<char>not_added_lower;
for(char x='a'; x<='z'; x++)
        if (find(unique.begin(), unique.end(),x)==unique.end())
                not_added_lower.push_back(x);
for(int i=0; i<not_added_lower.size(); i++){
        if ((last_col)%8==0){
                last_row+=1;
                last_col=0;
                mat[last_row][last_col++]=not_added_lower[i];
        }
        else mat[last_row][last_col++]=not_added_lower[i];
}
vector<char>not_added_upper;
for(char x='A'; x<='Z'; x++)
        if (find(unique.begin(), unique.end(),x)==unique.end())
                not_added_upper.push_back(x);
for(int i=0; i<not_added_upper.size(); i++){
        if ((last_col)%8==0){
                last_row+=1;
                last_col=0;
                mat[last_row][last_col++]=not_added_upper[i];
        }
        else mat[last_row][last_col++]=not_added_upper[i];
}
for (int i=0; i<10; i++){
        if ((last_col)%8==0){
```

```
                last_row+=1;
                last_col=0;
                mat[last_row][last_col++]=i+'0';
        }
        else mat[last_row][last_col++]=i+'0';
    }
    cout<<"\n";
    for(int i=0; i<8; i++){
        for(int j=0; j<8; j++)
            cout<<" "<<mat[i][j];
        cout<<endl;
    }
}
char encrypt(char a,char b,vector<char>& output){
    int x1,x2,y1,y2;
    for(int i=0; i<8; i++)
        for(int j=0; j<8; j++)
            if (mat[i][j]==a){
                x1=i;
                y1=j;
                break;
            }
    for(int i=0; i<8; i++)
        for(int j=0; j<8; j++)
            if (mat[i][j]==b){
                x2=i;
                y2=j;
                break;
            }
    if (x1==x2){
```

```
                output.push_back(mat[x1][(y1+1)%8]);
                output.push_back(mat[x2][(y2+1)%8]);
                output.push_back(' ');
        }
        else if (y1==y2){
                output.push_back(mat[(x1+1)%8][y1]);
                output.push_back(mat[(x2+1)%8][y2]);
                output.push_back(' ');
        }
        else{
                output.push_back(mat[x2][y1]);
                output.push_back(mat[x1][y2]);
                output.push_back(' ');
        }
}
void solve(vector<char> & input_1,vector<char> & input_2){
        vector<char>:: iterator it = input_1.begin();
        vector<char>:: iterator it2 = input_2.begin();
        vector<char> output;
        cout<<"\n The Decrypted Message: ";
        for(;it!=input_1.end();it++,it2++){
                cout<<*it<<*it2<<" ";
                encrypt(*it,*it2,output);
        }
        vector<char>:: iterator ot = output.begin();
        cout<<"\n The Encrypted Message: ";
        for(;ot!=output.end();ot++)
                cout<<*ot;
}
int main(){
```

```
string s1;
cout<<"\n Enter key string for the matrix: ";
cin >> s1;
build_matrix(s1,mat);
ifstream inf("input.txt");
char c,i1,i2;
vector<char>input_1;
vector<char>input_2;
while((c=inf.get())!=EOF){
        if(!isalnum(c))
        continue;
        i1=c;
        i2=inf.get();
        while(!isalnum(i2))
                i2=inf.get();
        if(i1==i2){
                input_1.push_back(c);
                input_2.push_back('!');
                input_2.push_back(c);
                input_1.push_back('!');
        }
        else{
                input_1.push_back(i1);
                input_2.push_back(i2);
        }
}
inf.close();
solve(input_1,input_2);
cout<<endl<<endl;
```

```
    return 0;
}
```

OUTPUT:

```
deepak@deepak:~/Desktop$ cat > input.txt
gdjagdjashdjas%^^76FHGFH
deepak@deepak:~/Desktop$ cat > input.txt
hhj
deepak@deepak:~/Desktop$ cat > input.txt
hkjhkk
deepak@deepak:~/Desktop$
```

```
deepak@deepak:~/Desktop$ ./play_fair

 Enter key string for the matrix: hjh7

 7 h j a b c d e
 f g i k l m n o
 p q r s t u v w
 x y z A B C D E
 F G H I J K L M
 N O P Q R S T U
 V W X Y Z 0 1 2
 3 4 5 6 7 8 9 !

 The Decrypted Message: hk jh k! !k
 The Encrypted Message: ga aj 6o o6

deepak@deepak:~/Desktop$
```

9. Program to implement Rail fence Cipher.

Program:

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>
void main(){
        int i,j=0,d,k=0;
        char p[50],ct[50][50];
        printf("Enter the plain text:\n");
        fgets(p,sizeof(p),stdin);
        printf("\nEnter the depth in the integer:");
        scanf("%d",&d);
        for(i=0;i<50;i++){
                for(j=0;j<50;j++){
                        ct[i][j]='\0';
                }
        }
        k=0;
        {
        for(i=0;i<strlen(p);i++){
                for(j=0;j<d;j++)
                {
                        if(k<=strlen(p))
                    ct[i][j]=p[k];
                        k++;
                }
                ct[i][j]='\0';
                }
```

```
        }

        for(i=0;i<d;i++){
                for(j=0;j<strlen(p);j++){
                        if(ct[j][i]!='\0') {
                            printf("%c",ct[j][i]);
                          }
                }
                printf("\n");
        }
        printf("\nThe encrypted text is:\n");
        for(i=0;i<d;i++)
        {
                for(j=0;j<strlen(p);j++)
                {
                        if(ct[j][i]!='\0')
                        printf("%c",ct[j][i]);
                }
        }
        getch();
}
```

Output:

```
deepak@deepak ~/cryptography $ ./a.out
Enter the plain text:
who are you

Enter the depth in the integer:3
w eo
ha u
ory


The encrypted text is:
w eoha uory
```

10.Program to implement simplified AES.

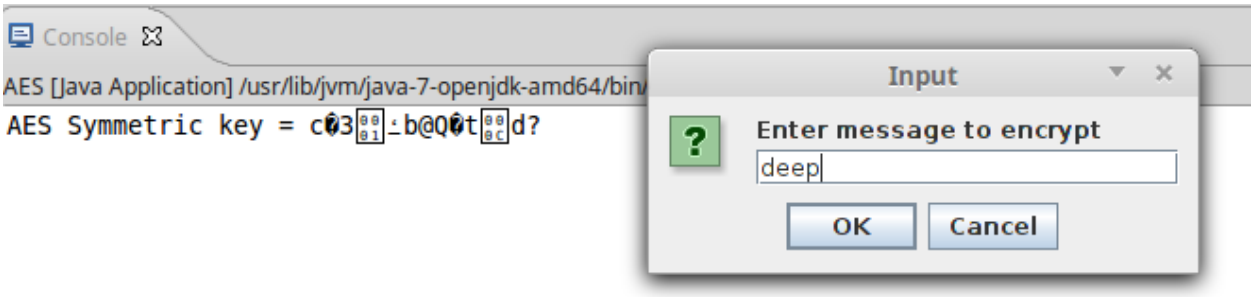
Program:

```
import javax.swing.*;
import java.security.SecureRandom;
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import java.util.Random ;


class AES {
	byte[] skey = new byte[1000];
	String skeyString;
	static byte[] raw;
	String inputMessage,encryptedData,decryptedMessage;

	public AES() {
		try {
		generateSymmetricKey();

		inputMessage=JOptionPane.showInputDialog(null,"Enter message to encrypt");
		byte[] ibyte = inputMessage.getBytes();
		byte[] ebyte=encrypt(raw, ibyte);
		String encryptedData = new String(ebyte);
		System.out.println("Encrypted message "+encryptedData);
		JOptionPane.showMessageDialog(null,"Encrypted Data "+"\n"+encryptedData);
```

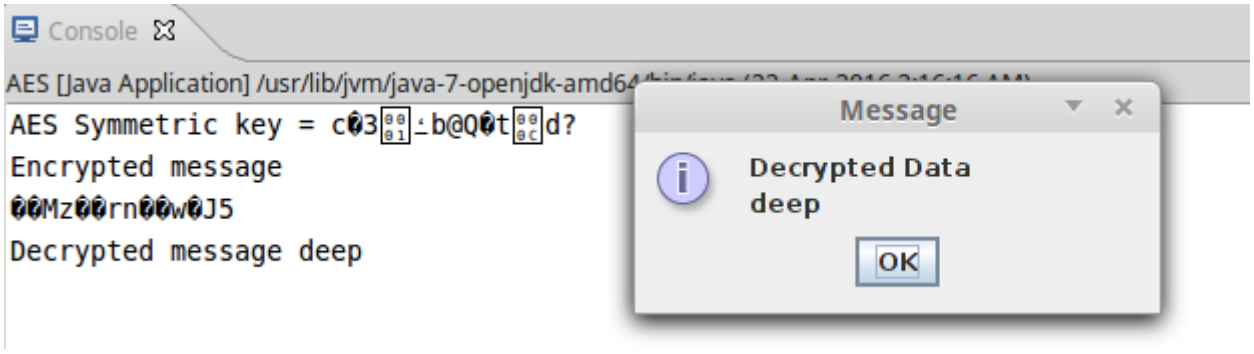
```
                byte[] dbyte= decrypt(raw,ebyte);
                String decryptedMessage = new String(dbyte);
                System.out.println("Decrypted message "+decryptedMessage);


                JOptionPane.showMessageDialog(null,"Decrypted Data
"+"\n"+decryptedMessage);
                }
                catch(Exception e) {
                        System.out.println(e);
                }


        }
   void generateSymmetricKey() {
      try {
                        Random r = new Random();
                        int num = r.nextInt(10000);
                        String knum = String.valueOf(num);
         byte[] knumb = knum.getBytes();
         skey=getRawKey(knumb);
         skeyString = new String(skey);
         System.out.println("AES Symmetric key = "+skeyString);
      }
      catch(Exception e) {
         System.out.println(e);
      }
   }
   private static byte[] getRawKey(byte[] seed) throws Exception {
      KeyGenerator kgen = KeyGenerator.getInstance("AES");
      SecureRandom sr = SecureRandom.getInstance("SHA1PRNG");
      sr.setSeed(seed);
```

```
        kgen.init(128, sr); // 192 and 256 bits may not be available
        SecretKey skey = kgen.generateKey();
        raw = skey.getEncoded();
        return raw;
    }
    private static byte[] encrypt(byte[] raw, byte[] clear) throws Exception {
        SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");
        Cipher cipher = Cipher.getInstance("AES");
        cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
        byte[] encrypted = cipher.doFinal(clear);
        return encrypted;
    }

    private static byte[] decrypt(byte[] raw, byte[] encrypted) throws Exception {
        SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");
        Cipher cipher = Cipher.getInstance("AES");
        cipher.init(Cipher.DECRYPT_MODE, skeySpec);
        byte[] decrypted = cipher.doFinal(encrypted);
        return decrypted;
    }


        public static void main(String args[]) {
                AES aes = new AES();
        }
}
```

Outputs:


Console
AES [Java Application] /usr/lib/jvm/java-7-openjdk-amd64/bin/
AES Symmetric key = c3 b@Q t d?
Input
Enter message to encrypt
deep
OK
Cancel


Console
AES [Java Application] /usr/lib/jvm/java-7-openjdk-amd64/b
AES Symmetric key = c3 b@Q t d?
Encrypted message
Mz rn w J5
Message
Encrypted Data
Mz rn w J5
OK


Console
AES [Java Application] /usr/lib/jvm/java-7-openjdk-amd64
AES Symmetric key = c3 b@Q t d?
Encrypted message
Mz rn w J5
Decrypted message deep
Message
Decrypted Data
deep
OK

11.Program to check primality using Miller – Rabin theorem.

Program:

```
#include<stdio.h>
int power(int x,int y)
{
	int i;
	int r=1;
	for(i=0;i<y;i++)
	{
		r=r*x;
	}
	return r;
}
void main(){
	int n,m,i;
	printf("\nenter a number");
	scanf("%d",&n);
	if(n%2==0)
		printf("\nnumber is composite");
	else{
		int k=0;
		m=n-1;
		while(m%2==0)
		{
			k++;
			m=m/2;
		}
		printf("m=%d,k=%d\n",m,k);
		int intermediate,power_of_2,product;
		printf("m=%dpowe=%d\n",m,power(10,m));
		intermediate=power(10,m)%n;
		product=intermediate*intermediate;
		printf("\nintermediate=%d",intermediate);
		for(i=1;i<k;i++)
		{
			intermediate=product%n;
			printf("\nintermediate=%d",intermediate);
			if(intermediate==n-1)
			{
				printf("\nnumber is prime\n");
				break;
			}
```

```
                product=intermediate*intermediate;
        }
        if(i==k)
                printf("\nnumber is composite\n");
    }
}
```

Output:

```
deepak@deepak ~/cryptography $ gcc miller_rabin.c
deepak@deepak ~/cryptography $ ./a.out

enter a number97
m=3,k=5

intermediate=30
intermediate=27
intermediate=50
intermediate=75
intermediate=96
number is prime
deepak@deepak ~/cryptography $
```

12.Program to solve the equations using Chinese Remainder theorem.

Program:

```
#include<stdio.h>
int power(int x,int y)
{
	int i;
	//printf("in power\n");
	int r=1;
	for(i=0;i<y;i++)
	{
		r=r*x;
	}
	return r;
}
void main(){
	int n,i,x;
	printf("\nenter no. of equations\n");
	scanf("%d",&n);
	int num1[n],mod[n],M[i],M_inverse[i];
	printf("\nenter equations as--> number(mod number)\n");
	for(i=0;i<n;i++)
	{
		scanf("%d%d",&num1[i],&mod[i]);
	}
	printf("\nequations are-->\n");
	for(i=0;i<n;i++)
	{
		printf("x = %d(mod %d)\n",num1[i],mod[i]);
	}
	int m=1;
	for(i=0;i<n;i++){
		m=m*mod[i];
	}
	for(i=0;i<n;i++){
		M[i]=m/mod[i];
	}
	for(i=0;i<n;i++){
		M_inverse[i]=(power(M[i],mod[i]-2))%mod[i];
	}
	x=0;
	for(i=0;i<n;i++){
		x+=num1[i]*M[i]*M_inverse[i];
```

```
        }
        printf("\n the value of x=%d",x);
}
```

Output:

```
deepak@deepak ~/cryptography $ gcc Chinese_Remainder_Theorem.c
deepak@deepak ~/cryptography $ ./a.out

enter no. of equations
2

enter equations as--> number(mod number)
2       3
4       5

equations are-->
x = 2(mod 3)
x = 4(mod 5)

 the value of x=44deepak@deepak ~/cryptography $
```

13.Program to encrypt and decrypt the text using DES.

Program:

```
 import java.io.*;
import java.lang.*;

class SDES
    {
     public int K1, K2;
     public static final int P10[] = { 3, 5, 2, 7, 4, 10, 1, 9, 8, 6};
     public static final int P10max = 10;
     public static final int P8[] = { 6, 3, 7, 4, 8, 5, 10, 9};
     public static final int P8max = 10;
     public static final int P4[] = { 2, 4, 3, 1};
     public static final int P4max = 4;
     public static final int IP[] = { 2, 6, 3, 1, 4, 8, 5, 7};
     public static final int IPmax = 8;
     public static final int IPI[] = { 4, 1, 3, 5, 7, 2, 8, 6};
     public static final int IPImax = 8;
     public static final int EP[] = { 4, 1, 2, 3, 2, 3, 4, 1};
     public static final int EPmax = 4;
     public static final int S0[][] = {{ 1, 0, 3, 2},{ 3, 2, 1, 0},{ 0, 2, 1,
                                        3},{ 3, 1, 3, 2}};
     public static final int S1[][] = {{ 0, 1, 2, 3},{ 2, 0, 1, 3},{ 3, 0, 1,
                                        2},{ 2, 1, 0, 3}};

    public static int permute( int x, int p[], int pmax)
    {
```

```
    int y = 0;
    for( int i = 0; i < p.length; ++i)
     {
       y <<= 1;
       y |= (x >> (pmax - p[i])) & 1;
     }
     return y;
    }

    public static int F( int R, int K)
    {
      int t = permute( R, EP, EPmax) ^ K;
      int t0 = (t >> 4) & 0xF;
      int t1 = t & 0xF;
      t0 = S0[ ((t0 & 0x8) >> 2) | (t0 & 1) ][ (t0 >> 1) & 0x3 ];
      t1 = S1[ ((t1 & 0x8) >> 2) | (t1 & 1) ][ (t1 >> 1) & 0x3 ];
       t = permute( (t0 << 2) | t1, P4, P4max);
     return t;

}

public static int fK( int m, int K)
    {
       int L = (m >> 4) & 0xF;
       int R = m & 0xF;
       return ((L ^ F(R,K)) << 4) | R;
    }

public static int SW( int x)
{
```

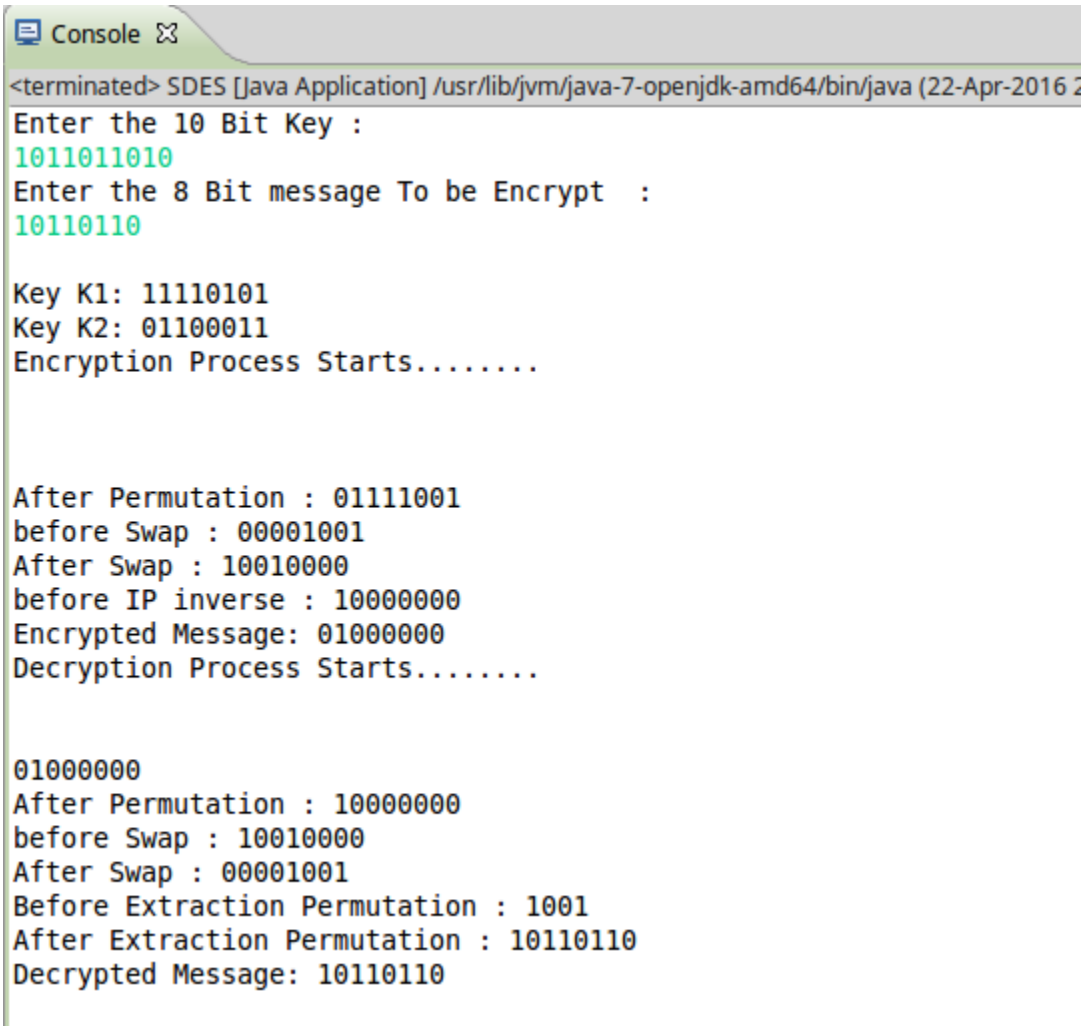
```
  return ((x & 0xF) << 4) | ((x >> 4) & 0xF);
 }



     public byte encrypt( int m)

     {
      System.out.println("\nEncryption Process Starts.....\n\n");
       m = permute( m, IP, IPmax);
      System.out.print("\nAfter Permutation : ");
       printData( m, 8);
       m = fK( m, K1);
       System.out.print("\nbefore Swap : ");
       printData( m, 8);
       m = SW( m);
       System.out.print("\nAfter Swap : ");
       printData( m, 8);
       m = fK( m, K2);
       System.out.print("\nbefore IP inverse : ");
       printData( m, 8);
       m = permute( m, IPI, IPImax);
       return (byte) m;

     }



     public byte decrypt( int m)

     {
       System.out.println("\nDecryption Process Starts.....\n\n");
```

```
  printData( m, 8);
  m = permute( m, IP, IPmax);
  System.out.print("\nAfter Permutation : ");
  printData( m, 8);
  m = fK( m, K2);
  System.out.print("\nbefore Swap : ");
  printData( m, 8);
  m = SW( m);
  System.out.print("\nAfter Swap : ");
  printData( m, 8);
  m = fK( m, K1);
  System.out.print("\nBefore Extraction Permutation : ");
  printData( m, 4);
  m = permute( m, IPI, IPImax);
  System.out.print("\nAfter Extraction Permutation : ");
  printData( m, 8);
  return (byte) m;

}

public static void printData( int x, int n)
 {
   int mask = 1 << (n-1);
   while( mask > 0)
   {
   System.out.print( ((x & mask) == 0) ? '0' : '1');
   mask >>= 1;
   }
}
```

```
 public SDES( int K)
{
   K = permute( K, P10, P10max);
   int t1 = (K >> 5) & 0x1F;
   int t2 = K & 0x1F;
   t1 = ((t1 & 0xF) << 1) | ((t1 & 0x10) >> 4);
   t2 = ((t2 & 0xF) << 1) | ((t2 & 0x10) >> 4);
   K1 = permute( (t1 << 5)| t2, P8, P8max);
   t1 = ((t1 & 0x7) << 2) | ((t1 & 0x18) >> 3);
   t2 = ((t2 & 0x7) << 2) | ((t2 & 0x18) >> 3);
   K2 = permute( (t1 << 5)| t2, P8, P8max);

 }



 public static void main( String args[]) throws Exception
 {
  DataInputStream inp=new DataInputStream(System.in);
  System.out.println("Enter the 10 Bit Key :");
   int K = Integer.parseInt(inp.readLine(),2);
   SDES A = new SDES( K);
   System.out.println("Enter the 8 Bit message To be Encrypt  : ");
   int m = Integer.parseInt(inp.readLine(),2);
   System.out.print("\nKey K1: ");
   SDES.printData( A.K1, 8);
   System.out.print("\nKey K2: ");
```

```
        SDES.printData( A.K2, 8);
        m = A.encrypt( m);
        System.out.print("\nEncrypted Message: ");
        SDES.printData( m, 8);
        m = A.decrypt( m);
        System.out.print("\nDecrypted Message: ");
        SDES.printData( m, 8);
} }
```

Output:


Console
<terminated> SDES [Java Application] /usr/lib/jvm/java-7-openjdk-amd64/bin/java (22-Apr-2016 2
Enter the 10 Bit Key :
1011011010
Enter the 8 Bit message To be Encrypt :
10110110
Key K1: 11110101
Key K2: 01100011
Encryption Process Starts........
After Permutation : 01111001
before Swap : 00001001
After Swap : 10010000
before IP inverse : 10000000
Encrypted Message: 01000000
Decryption Process Starts........
01000000
After Permutation : 10000000
before Swap : 10010000
After Swap : 00001001
Before Extraction Permutation : 1001
After Extraction Permutation : 10110110
Decrypted Message: 10110110

14.Program to implement fast exponentiation.

Program:

```
#include<stdio.h>
long squareOfMultiply(long a,long x,long n){
	int i,k=0;
	long y=1,t,binary[100];
	long num=x;
	while(num!=0)
	{
		binary[k]=num%2;
		k++;
		num=num/2;
	}
	for(i=0;i<k;i++)
	{
		if(binary[i]==1){
			y=(a*y)%n;
			a=(a*a)%n;
		}
		else
		{
			a=(a*a)%n;
		}
	}
	return y;
}
void main(){
	long a,m,n;
	printf("enter value of a,m and n in-------->  a^m mod n");
	scanf("%ld %ld %ld",&a,&m,&n);
	long s=squareOfMultiply(a,m,n);
	printf("\nvalue of a^m mod n = %ld",s);
}
```

Output:

```
deepak@deepak ~/cryptography $ gcc fast_exponentiation.c
deepak@deepak ~/cryptography $ ./a.out
enter value of a,m and n in-------->  a^m mod n
9
43
77

value of a^m mod n = 58deepak@deepak ~/cryptography $
```

15.Program to implement RSA algorithm.

Program:

```
#include<stdio.h>
int power(int x,int y)
{
	int i;
	//printf("in power\n");
	int r=1;
	for(i=0;i<y;i++)
	{
		r=r*x;
	}
	return r;
}
int bits(int x)
{
	int s=0;
	while(x)
	{
		x=x/10;
		s++;
	}
	return s;
}
int squareOfMultiply(int a,int x,int n)
{
	int i,k=0,y=1,t,binary[100];
	int num=x;
	while(num!=0)
	{
		binary[k]=num%2;
		//r=r+t*power(10,i);
		k++;
		num=num/2;
	}
	//printf("\nBinary of x\n");
	for(i=0;i<k;i++)
	{
		//printf("%d  ",binary[i] );
	}
	//printf("\nx=%d,bits=%d\n",x,k);
	for(i=0;i<k;i++)
	{
```

```
                if(binary[i]==1){
                        y=(a*y)%n;
                        a=(a*a)%n;
                        //printf("\nhere---i=%d,binary[i]=%d,y=%d,a=%d\n",i,binary[i],y,a );
                }
                else
                {
                        a=(a*a)%n;
                        //printf("\ni=%d,binary[i]=%d,y=%d,a=%d\n",i,binary[i],y,a );
                }
        }
        return y;
        //printf("\ny=%d",y);
}
void main()
{
        //printf("Enter (a,x,n) ----->a^x mod n\n");
        //int a=17,x=22,n=21;
        //printf("\n Entered value is---> %d^%d mod %d",a,x,n);
        //int y=squareOfMultiply(a,x,n);
        //printf("y=%d\n",y);
        int p=7,q=11;
        //printf("\nenter any two prime numbers p and q where p!=q");
        //scanf("%d%d",&p,&q);
        printf("\nvalue of p=%d,q=%d",p,q);
        int n=p*q;
        int phi_of_n=(p-1)*(q-1);
        printf("\nphi(n)=%d",phi_of_n);
        int e=13;
        //int d=(power(e,phi_of_n-2))%phi_of_n;
        int d=37;
        printf("\nd=%d",d);
        int PT=5;
        //printf("\nEnter thePlainText");
        //scanf("%d",PT);
        printf("\nEntered value is---> %d^%d mod %d",PT,e,n);
        int y=squareOfMultiply(PT,e,n);
        printf("\ny=%d\n",y);
}
```

Output:

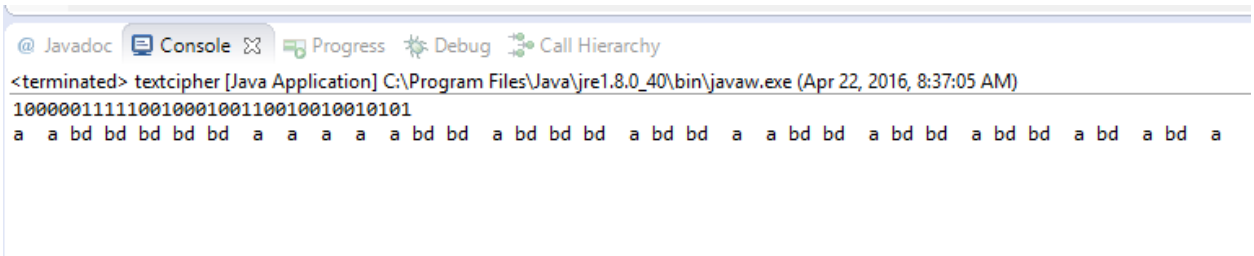
```
deepak@deepak ~/cryptography $ gcc RSA.c
deepak@deepak ~/cryptography $ ./a.out

value of p=7,q=11
phi(n)=60
d=37
Entered value is---> 5^13 mod 77
y=26
deepak@deepak ~/cryptography $
```

16.Program to implement text cover.

Program:

```
package servlet;
import java.util.Scanner;
public class textcipher {
	public static void main(String args[])
	{//Input the text that has to be encrypted
		Scanner obj=new Scanner(System.in);
		String Key="AOHIT";String bin="";String fin="";int h;
		char c[]=Key.toCharArray();
		int l=Key.length();
		char ch;int conv;
		for(int i=0;i<l;i++){
			ch=c[i];
			conv=(int)ch;
			while(conv>=1){
				h=conv%2;
				conv=conv/2;
				bin=bin+h;
		}
	}
		System.out.println(bin);
		int k=bin.length();
		char d[]=bin.toCharArray();
		String enc="a";
		for(int j=0;j<k;j++){
			if(d[j]=='1')
				enc=enc+"  a";
			else
				enc=enc+" bd";
		}
		System.out.println(enc);
}}
```

Output:



17.Program to implement random number generator.

Program:

```
#include <stdio.h>

int jsw_lcg(int seed)
{
   return  (2 * seed + 3) % 10;
}

int main(void)
{
   int seed;
   int i;

   printf("\nEnter seed value");
   scanf("%d",&seed);
   for (i = 0; i < 10; i++)
   {
      printf("%d ", seed);
      seed = jsw_lcg(seed);
   }

   printf("...\n");

   return 0;
}
```

Output:

```
deepak@deepak ~/cryptography $ gcc random_number_generator.c
deepak@deepak ~/cryptography $ ./a.out

Enter seed value5
5 3 9 1 5 3 9 1 5 3 ...
deepak@deepak ~/cryptography $
```

18.Program to implement Discrete algorithm.

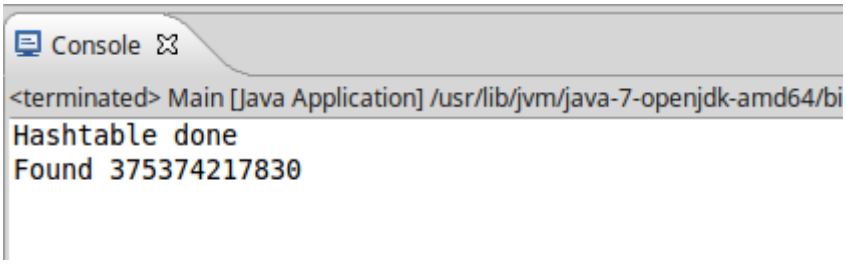
Program:

```
import java.math.BigInteger;
 import java.util.HashMap;
 import java.util.Map;



 public class Main {


     //sample numbers. Note we MUST use BigIntegers
     static BigInteger h = new
BigInteger("32394751040504504435652643787280657886490975209524495278347924529719819761432925580738569379585531805328789280014947060973941085775857324523076734440203333");
     static BigInteger g = new
BigInteger("11717829880366207009516117596335367088558084999998952205599979459063929499736583746670572176471460312928594829675428279466566527115212748467589894601965568");
     static BigInteger p = new
BigInteger("13407807929942597099574024998205846127479365820592393377723561443721764030073546976801874298166903427690031858186486050853753882811946569946433649006084171");
     static long B = 1048576;//2^20


     //build hashtable of all possible h/(g^x1) for x1 in 0..B
     private static Map<BigInteger, Long> leftHash(){
         Map<BigInteger, Long> m = new HashMap<BigInteger, Long>();
         BigInteger n, gpow, ginversepow;
         for(long i=0; i<B; i++){
             //compute g^x1 mod p
             gpow = g.modPow(new BigInteger(i+""), p);
```

```
            //compute 1/(g^x1) mod p
            ginversepow = gpow.modInverse(p);
            //compute h/(g^x1) mod p
            n = h.multiply(ginversepow);
            n = n.mod(p);
            //store in hashtable
            m.put(n, i);
        }
        System.out.println("Hashtable done");
        return m;
    }

    //compute n = g^B^x0 for x0 in 0..B, then check if n is in hashtable. If it is, we found (x0, x1) and can compute x as x0*B+x1
    private static long computeDiscreteLog(Map<BigInteger, Long> m){
        BigInteger n;
        long res = 0;
        //compute g^B
        BigInteger gB = g.modPow(new BigInteger(B+""), p);
        for(long i=0; i<B; i++){
            //compute g^B^x0
            n = gB.modPow(new BigInteger(i+""), p);
            if(m.containsKey(n)){
                res = i*B+m.get(n);
                break;
            }
        }
        return res;
    }
```

```
    public static void main(String [] args){
        Map<BigInteger, Long> m = leftHash();
        long res = computeDiscreteLog(m);
        System.out.println("Found "+res);
    }

}
```

Output:



19.Program to implement DSA algorithm.

Program:

```
#include<stdio.h>
long power(long x,long y)
{
	int i;
	long r=1;
	for(i=0;i<y;i++)
	{
		r=r*x;
	}
	return r;
}
int modInverse(int a, int m)
{
	int m0 = m, t, q;
	int x0 = 0, x1 = 1;

	if (m == 1)
	return 0;

	while (a > 1)
	{
		// q is quotient
		q = a / m;

		t = m;

		// m is remainder now, process same as
		// Euclid's algo
		m = a % m, a = t;

		t = x0;

		x0 = x1 - q * x0;
			x1 = t;
	}

	// Make x1 positive
	if (x1 < 0)
	x1 += m0;

	return x1;
}
```

```
long squareOfMultiply(long a,long x,long n)
{
	int i,k=0;
	long y=1,t,binary[100];
	long num=x;
	while(num!=0)
	{
		binary[k]=num%2;
		k++;
		num=num/2;
	}
	for(i=0;i<k;i++)
	{
		if(binary[i]==1){
			y=(a*y)%n;
			a=(a*a)%n;
		}
		else
		{
			a=(a*a)%n;
		}
	}
	return y;
}
long m=1;
long mod(long a, long n){
	if(a>0){
		return a%n;
	}
	else{
		a=a+m*n;
		m++;
		if(a<0) mod(a,n);
		else	return a;
	}
}
void main(){
	long p=23,q=11,a=5,g=2,d=8,h=12,k=5;
	long g_pow_d=squareOfMultiply(g,d,p);
	printf("\n----------------------Alice Key Generation----------------------");
	printf("\nAlice public key (p,q,g,g_pow_d) = (%ld,%ld,%ld,%ld)",p,q,g,g_pow_d);
	printf("\n----------------------Signing by Alice----------------------------");
	long r=squareOfMultiply(g,k,p);
	r=r%q;
	long s=(h+d*r);
	long x=modInverse(k,q);
```

```
        s=(s*x)%q;
        printf("\n value of r=%ld s=%ld",r,s);
        printf("\n----------Verification by Bob----------");
        x=modInverse(s,q);
        long u1=(h*x)%q;
        long u2=(r*x)%q;
        printf("\nvalue of u1=%ld u2=%ld",u1,u2);
        x=power(g,u1)%p;
        long y=power(g_pow_d,u2)%p;
        x=(x*y)%p;
        long v=x%q;
        printf("\nvalue of v=%ld\n",v);
}
```

Output:

```
deepak@deepak ~/cryptography $ ./a.out

----------Alice Key Generation----------
Alice public key (p,q,g,g_pow_d) = (23,11,2,3)
----------Signing by Alice----------
 value of r=9 s=8
----------Verification by Bob----------
value of u1=7 u2=8
value of v=9
deepak@deepak ~/cryptography $
```

20.Program to implement Elgamal DSA.

Program:

```
#include<stdio.h>
long t1=0,t2=1,r,q,t;
long squareOfMultiply(long a,long x,long n)
{
	int i,k=0;
	long y=1,t,binary[100];
	long num=x;
	while(num!=0)
	{
		binary[k]=num%2;
		k++;
		num=num/2;
	}
	for(i=0;i<k;i++)
	{
		if(binary[i]==1){
			y=(a*y)%n;
			a=(a*a)%n;
		}
		else
		{
			a=(a*a)%n;
		}
	}
	return y;
}
long reverse(long r1,long r2){
	if(r1<=1)
		return t1;
	else
	{
		//printf("\nvalue of r=%ld, q=%ld, r1=%ld, r2=%ld, t=%ld, t1=%ld,
t2=%ld",r,q,r1,r2,t,t1,t2);
		r=r1%r2;
		q=r1/r2;
		r1=r2;
		r2=r;
		t=t1-t2*q;
		t1=t2;
		t2=t;

		reverse(r1,r2);
```

```
        }

}
long m=1;
long mod(long a, long n){
        if(a>0){
                return a%n;
        }
        else{
                a=a+m*n;
                m++;
                if(a<0) mod(a,n);
                else     return a;
        }
}
long power(long x,long y)
{
        long i;
        //printf("in power\n");
        long r=1;
        for(i=0;i<y;i++)
        {
                r=r*x;
        }
        return r;
}
void main(){
        long p=23,g=5,d=3,k=9,h=7;
        long g_pow_d=squareOfMultiply(g,d,p);
        printf("\n-----------Key Generation-----------");
        printf("\nPublic Key (p,g,g_pow_d) = (%ld,%ld,%ld)",p,g,g_pow_d);
        printf("\n-----------Signing-----------------");
        long r=squareOfMultiply(g,k,p);
        long x=reverse(p-1,k);
        x=(h-d*r)*x;
        long s=mod(x,p-1);
        printf("\nvalue of (r,s) = (%ld,%ld)",r,s);
        printf("\n---------Verification--------------");
        long v1=squareOfMultiply(g,h,p);
        x=power(g_pow_d,r);
        x=power(r,s)*x;
        long v2=mod(x,p);
        printf("\nvalue of (v1,v2) = (%ld,%ld)\n",v1,v2);
}
```

Output:

```
deepak@deepak ~/cryptography $ gcc Elgamal_DSA.c
deepak@deepak ~/cryptography $ ./a.out

-----------Key Generation-----------
Public Key (p,g,g_pow_d) = (23,5,10)
-----------Signing------------------
value of (r,s) = (11,2)
---------Verification--------------
value of (v1,v2) = (17,17)
deepak@deepak ~/cryptography $
```

21.Program to implement RSA DSA.

Program:

```
#include<stdio.h>
long t1=0,t2=1,r,q,t;
long reverse(long r1,long r2){
	if(r1<=1)
		return t1;
	else
	{
		//printf("\nvalue of r=%ld, q=%ld, r1=%ld, r2=%ld, t=%ld, t1=%ld,
t2=%ld",r,q,r1,r2,t,t1,t2);
		r=r1%r2;
		q=r1/r2;
		r1=r2;
		r2=r;
		t=t1-t2*q;
		t1=t2;
		t2=t;

		reverse(r1,r2);
	}

}
long squareOfMultiply(long a,long x,long n)
{
	int i,k=0;
	long y=1,t,binary[100];
	long num=x;
	while(num!=0)
	{
		binary[k]=num%2;
		k++;
		num=num/2;
	}
	for(i=0;i<k;i++)
	{
		if(binary[i]==1){
			y=(a*y)%n;
			a=(a*a)%n;
		}
		else
		{
			a=(a*a)%n;
		}
```

```
        }
        return y;
}
void main(){
        long p=7,q=11,e=7,m=9;
        long n=p*q;
        long phi=(p-1)*(q-1);
        long d;
        long k=reverse(phi,e);
        if(k<0) d=k+phi;
        //printf("\nphi=%ld e=%ld",phi,e);
        printf("\n-------Key generation-------");
        printf("\npublic key------>(e,n)=(%ld,%ld)",e,n);
        printf("\nprivate key------>(d,n)=(%ld,%ld)",d,n);
        printf("\n-------Signing---------");
        long s=squareOfMultiply(m,d,n);
        printf("\nsignature s=%ld",s);
        printf("\n------Verification------");
        long x=squareOfMultiply(s,e,n);
        printf("\n value of x=%ld\n",x);
}
```

Output:

```
deepak@deepak ~/cryptography $ gcc RSA_DSA.c
deepak@deepak ~/cryptography $ ./a.out

-------Key generation-------
public key------>(e,n)=(7,77)
private key------>(d,n)=(43,77)
-------Signing---------
signature s=58
------Verification------
 value of x=9
deepak@deepak ~/cryptography $
```

22.Program to implement Diffie Hellman algorithm.

Program:

```
#include<stdio.h>
long squareOfMultiply(long a,long x,long n)
{
	int i,k=0;
	long y=1,t,binary[100];
	long num=x;
	while(num!=0)
	{
		binary[k]=num%2;
		k++;
		num=num/2;
	}
	for(i=0;i<k;i++)
	{
		if(binary[i]==1){
			y=(a*y)%n;
			a=(a*a)%n;
		}
		else
		{
			a=(a*a)%n;
		}
	}
	return y;
}
void main(){
	long g=7,p=23,x=3,y=6;
	printf("\nValue of g=%ld, p=%ld",g,p);
	printf("\n----------Alice calcutes R1----------");
	long R1=squareOfMultiply(g,x,p);
	printf("\nx=%ld , R1=%ld",x,R1);
	printf("\n----------Bob calcutes R2----------");
	long R2=squareOfMultiply(g,y,p);
	printf("\ny=%ld , R2=%ld",y,R2);
	printf("\nAlice sends R1 = %ld to Bob",R1);
	printf("\nBob sends R2 = %ld to Alice",R2);
	printf("\n----------Verification----------");
	long k=squareOfMultiply(R2,x,p);
	printf("\nAlice calculates k = %ld",k);
	k=squareOfMultiply(R1,y,p);
	printf("\nBob calculates k = %ld\n",k);
}
```

Output:

```
deepak@deepak ~/cryptography $ gcc Diffie_Hellman.c
deepak@deepak ~/cryptography $ ./a.out

Value of g=7, p=23
----------Alice calcutes R1-----------
x=3 , R1=21
----------Bob calcutes R2-------------
y=6 , R2=4
Alice sends R1 = 21 to Bob
Bob sends R2 = 4 to Alice
-----------Verification--------------
Alice calculates k = 18
Bob calculates k = 18
deepak@deepak ~/cryptography $
```