APPLIED CRYPTOGRAPHY

LAB MANUAL

CONTENTS

XOR

AIM :

Write a C program that contains a string (char pointer) with a value 'Hello World'. The program should XOR each character in this string with 0 and displays the result.

ALGORITHM :

1. Declare a char pointer variable str and initialize it with the value "Hello World".
2. Loop through the string using a for loop.
3. For each character in the string, XOR it with 0 using the ^ operator.
4. Display the result using printf.

RESULT :

THE OUTPUT HAS BEEN OBTAINED SUCESSFULLY.

PROGRAM

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<string.h>
void main()
{
clrscr();
char str[]="Hello World";
char str1[11];
int i,len;
len=strlen(str);
for(i=0;i<len;i++)
{
str1[i]=str[i]^0;
printf("%c",str1[i]);
}
printf("\n");
getch(); }
```

OUTPUT

Hello World

AND or and XOR

AIM:

Write a C program that contains a string (char pointer) with a value \Hello World'. The program should AND or and XOR each character in this string with 127 and display the result.

ALGORITHM :

1. Declare a char pointer variable "str" and initialize it with the value "Hello World".
2. Loop through the string using a for loop.
3. For each character in the string, perform either the AND or XOR operation with 127 based on the desired output.
4. Display the result using printf.

RESULT :

THE OUTPUT HAS BEEN OBTAINED SUCESSFULLY.

PROGRAM

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<stdlib.h>
void main()
{
char str[]="Hello World";
char str1[11];
char str2[11];
int i,len;
len = strlen(str);
clrscr();
for(i=0;i<len;i++)
{
str1[i] = str[i]&127;
printf("%c",str1[i]);
}
printf("\n");
for(i=0;i<len;i++)
{
str2[i] = str[i]^127;
printf("%c",str2[i]);
}
printf("\n");
getch(); }
```

OUTPUT

ENCRYPTION AND DECRYPTION

AIM :

To Perform Encryption and Decryption using the following Algorithms.

CEASER CIPHER

ALGORITHM :

1. Accept the plaintext message as input from the user.
2. Accept the shift amount (the number of positions to shift each character in the plaintext) as input from the user.
3. Loop through each character in the plaintext message.
4. For each character, shift it by the shift amount using the Caesar Cipher formula:
5. If the character is an uppercase letter, add the shift amount modulo 26 to the ASCII value of the character.

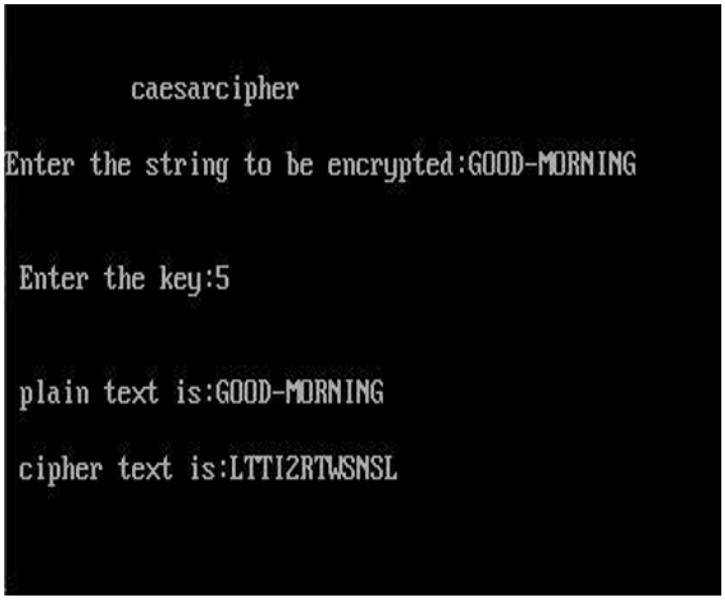
6. If the character is a lowercase letter, add the shift amount modulo 26 to the ASCII value of the character.
7. If the character is not a letter, leave it as is.
8. Concatenate the shifted characters to form the ciphertext message.
9. Display the ciphertext message.

PROGRAM

```
#include<iostream.h>
#include<conio.h>
void main()
{
char str[20],out[20];
int k;
int nkey[10];
clrscr();
cout<<"\n\n\t caesarcipher";
cout<<"\n\nEnter the string to be encrypted:";
cin>>str;
cout<<"\n\n Enter the key:";
cin>>k;
for(int i=0,j=0;str[i]!='\0';i++,j++)
```

```
out[j]=str[i]+k;
out[j]='\0';
cout<<"\n\n plain text is:"<<str;
cout<<"\n\n cipher text is:"<<out;
getch();
}
```

OUTPUT



SUBSTITUTION CIPHER

ALGORITHM :

1. Create a key for the substitution cipher. The key should be a mapping of each letter in the alphabet to a unique letter in the same alphabet. For example, 'a' could be mapped to 'q', 'b' to 'p', and so on.
2. Take the plaintext input to be encrypted.
3. Loop through each character in the plaintext input.
4. Use the substitution key to replace the current character with its corresponding letter.
5. Append the encrypted character to a new string that will hold the ciphertext.
6. Repeat steps 3-5 for all characters in the plaintext input.
7. Return the ciphertext.

PROGRAM

```
#include<iostream.h>
#include<conio.h>
#include<ctype.h> void main()
{
char str[20],out[20],key[10];
```

```
int nkey[10];
clrscr();
cout<<"\n\n\n polyalphabetic cipher";
cout<<"\n\n Enter the string to be encrypted:";
cin>>str;
cout<<"\n\nenter the key:";
cin>>key;
for(int m=0;key[m]!='\0';m++)
nkey[m]=(char)key[m]-97;
for(int i=0,j=0,k=0;str[i]!='\0';i++,j++,k++)
{
if(nkey[k]>=m) k=0;
if((str[i]+nkey[k])>=122)
out[j]=(str[i]+nkey[k])-26; else
out[j]=str[i]+nkey[k];
}
out[j]='\0';
cout<<"\n\n plaintext is :"<<str;
cout<<"\n\n cipher text is:"<<out;
getch();
}
```

OUTPUT

HILL CIPHER

ALGORITHM :

1. Define the key matrix with a dimension of n x n, where n is the size of the plaintext blocks to be encrypted.
2. Ensure the key matrix is invertible. If not, choose a different key.
3. Map each character in the plaintext to a number. For example, 'A' could be mapped to 0, 'B' to 1, and so on.
4. Group the plaintext characters into blocks of size n. If the last block is not complete, add padding characters.
5. For each plaintext block, multiply it by the key matrix to get the corresponding ciphertext block.
6. Map each number in the ciphertext block back to a character using the inverse of the mapping in step3.

RESULT :

THE OUTPUT HAS BEEN OBTAINED SUCESSFULLY.

PROGRAM

```
import java.io.*;
import java.util.*;
import java.io.*;
public class Hill
{
static float[][] decrypt = new float[3][1];
static float[][] a = new float[3][3];
static float[][] b = new float[3][3];
static float[][] mes = new float[3][1];
static float[][] res = new float[3][1];
static BufferedReader br = new BufferedReader(new
InputStreamReader(System.in)); static Scanner sc = new
Scanner(System.in);
public static void main(String[] args) throws IOException
{
getkeymes();
for(int i=0;i<3;i++)
for(int j=0;j<1;j++)
for(int k=0;k<3;k++)
{
res[i][j]=res[i][j]+a[i][k]*mes[k][j];
```

```
}
System.out.print("\nEncrypted string is :");
for(int i=0;i<3;i++)
{
System.out.print((char)(res[i][0]%26+97));
res[i][0]=res[i][0];
}
inverse();
for(int i=0;i<3;i++)
for(int j=0;j<1;j++)
for(int k=0;k<3;k++)
{
decrypt[i][j] = decrypt[i][j]+b[i][k]*res[k][j];

}
System.out.print("\nDecrypted string is : ");
for(int i=0;i<3;i++){
System.out.print((char)(decrypt[i][0]%26+97));
}
System.out.print("\n");
```

```
}
public static void getkeymes() throws IOException {
System.out.println("Enter 3x3 matrix for key (It should be inversible): ");
for(int i=0;i<3;i++)
for(int j=0;j<3;j++)
a[i][j] = sc.nextFloat();
System.out.print("\nEnter a 3 letter string: ");
String msg = br.readLine();
for(int i=0;i<3;i++)
mes[i][0] = msg.charAt(i)-97;
}
public static void inverse() {
float p,q;
float[][] c = a;
for(int i=0;i<3;i++)
for(int j=0;j<3;j++) {
if(i==j)
b[i][j]=1;
else b[i][j]=0;
```

```
}
for(int k=0;k<3;k++)
{
for(int i=0;i<3;i++) {
p = c[i][k];
q = c[k][k];
for(int 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(int i=0;i<3;i++)
for(int j=0;j<3;j++) {
b[i][j] = b[i][j]/c[i][i]; }
System.out.println("");
System.out.println("\nInverse Matrix is : ");
for(int i=0;i<3;i++) {
for(int j=0;j<3;j++)
System.out.print(b[i][j] + " ");
System.out.print("\n");}}}
```

OUTPUT

```
debug:
Enter 3x3 matrix for key (It should be inversible):
1
5
9
8
7
5
6
2
4

Enter a 3 letter string: hai

Encrypted string is :bsw

Inverse Matrix is :
-0.07964602 0.0088495575 0.16814159
0.0088495575 0.22123894 -0.29646018
0.11504425 -0.123893805 0.1460177

Decrypted string is : hai
BUILD SUCCESSFUL (total time: 1 minute 2 seconds)
```

DES

AIM :

To implement Data Encryption Standard Algorithm Using C/Java.

ALGORITHM :

1. Get plain text from the user.
2. Covert the plain text into its binary equivalent
3. Write a function IP that takes the binary string of plain text as input and outputs permuted form of the input.
4. Divide the output of the IP function into two binary strings each of 32 bits length. Step 5: Write a function EXPANSION that takes the second half of the output of the IP
5. function returns a 48 bit output.

6. Write a function EXOR that does the exor operation between the output of the expansion function and the 48 bit key generated.
7. Write a function SBOX that takes the 48bit output of the EXOR function and gives a 32 bit output using 8 standard s boxes.
8. Pass the output of the SBOX function and the first half of the output of the IP function to the EXOR function and get a 32 bit output.
9. Concatenate the input of the Expansion function and the output of the Step 8. This gives a 64 bit output which acts as input to the next round.
10. Repeat step 5 to 9 15 times and give the output of the 16th round to function IP_INVERSE that gives a 64 bit cipher text.

RESULT :

THE OUTPUT HAS BEEN OBTAINED SUCESSFULLY.

PROGRAM

```
package Dess;
import java.util.*;
class DES {
private static final byte[] 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
};
private static final byte[] 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
};
private static final byte[] 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
};
private static final byte[] rotations = {
1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1
};
private static final byte[] 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
};
private static final byte[][] S = { {
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
}, {
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
}, {
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
}, {
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

}, {

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

}, {

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

}, {

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

}, {

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

```
} };
private static final byte[] 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
};
private static final byte[] 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
};
private static int[] C = new int[28];
```

```
private static int[] D = new int[28];

private static int[][] subkey = new int[16][48];

public static void main(String args[]) {

System.out.println("Enter the input as a 16 character
hexadecimal value:");

String input = new Scanner(System.in).nextLine();

int inputBits[] = new int[64];

for(int i=0 ; i < 16 ; i++) {

String s =
Integer.toBinaryString(Integer.parseInt(input.charAt(i) +
"", 16));

while(s.length() < 4) {

s = "0" + s;

}

for(int j=0 ; j < 4 ; j++) {

inputBits[(4*i)+j] = Integer.parseInt(s.charAt(j) + "");

}

}

System.out.println("Enter the key as a 16 character
hexadecimal value:");

String key = new Scanner(System.in).nextLine();

int keyBits[] = new int[64];

for(int i=0 ; i < 16 ; i++) {
```

```
String s =
Integer.toBinaryString(Integer.parseInt(key.charAt(i) +
"", 16));
while(s.length() < 4) {
s = "0" + s;
}
for(int j=0 ; j < 4 ; j++) {
keyBits[(4*i)+j] = Integer.parseInt(s.charAt(j) + "");
}
}
System.out.println("\n+++ ENCRYPTION +++");
int outputBits[] = permute(inputBits, keyBits, false);
System.out.println("\n+++ DECRYPTION +++");
permute(outputBits, keyBits, true);
}
private static int[] permute(int[] inputBits, int[] keyBits,
boolean isDecrypt) {
int newBits[] = new int[inputBits.length];
for(int i=0 ; i < inputBits.length ; i++) {
newBits[i] = inputBits[IP[i]-1];
}
int L[] = new int[32];
int R[] = new int[32];
```

```
int i;
for(i=0 ; i < 28 ; i++) {
C[i] = keyBits[PC1[i]-1];
}
for( ; i < 56 ; i++) {
D[i-28] = keyBits[PC1[i]-1];
}
System.arraycopy(newBits, 0, L, 0, 32);
System.arraycopy(newBits, 32, R, 0, 32);
System.out.print("\nL0 = ");
displayBits(L);
System.out.print("R0 = ");
displayBits(R);
for(int n=0 ; n < 16 ; n++) {
System.out.println("\n------------");
System.out.println("Round " + (n+1) + ":");
int newR[] = new int[0];
if(isDecrypt) {
newR = fiestel(R, subkey[15-n]);
System.out.print("Round key = ");
displayBits(subkey[15-n]);
} else {
```

```
newR = fiestel(R, KS(n, keyBits));
System.out.print("Round key = ");
displayBits(subkey[n]);
}
int newL[] = xor(L, newR);
L = R;
R = newL;
System.out.print("L = ");
displayBits(L);
System.out.print("R = ");
displayBits(R);
}
int output[] = new int[64];
System.arraycopy(R, 0, output, 0, 32);
System.arraycopy(L, 0, output, 32, 32);
int finalOutput[] = new int[64];
for(i=0 ; i < 64 ; i++) {
finalOutput[i] = output[FP[i]-1];
}
String hex = new String();
for(i=0 ; i < 16 ; i++) {
String bin = new String();
```

```
for(int j=0 ; j < 4 ; j++) {
bin += finalOutput[(4*i)+j];
}
int decimal = Integer.parseInt(bin, 2);
hex += Integer.toHexString(decimal);
}
if(isDecrypt) {
System.out.print("Decrypted text: ");
} else {
System.out.print("Encrypted text: ");
}
System.out.println(hex.toUpperCase());
return finalOutput;
}
private static int[] KS(int round, int[] key) {
int C1[] = new int[28];
int D1[] = new int[28];
int rotationTimes = (int) rotations[round];
C1 = leftShift(C, rotationTimes);
D1 = leftShift(D, rotationTimes);
int CnDn[] = new int[56];
System.arraycopy(C1, 0, CnDn, 0, 28);
```

```
System.arraycopy(D1, 0, CnDn, 28, 28);
int Kn[] = new int[48];
for(int i=0 ; i < Kn.length ; i++) {
Kn[i] = CnDn[PC2[i]-1];
}
subkey[round] = Kn;
C = C1;
D = D1;
return Kn;
}
private static int[] fiestel(int[] R, int[] roundKey) {
int expandedR[] = new int[48];
for(int i=0 ; i < 48 ; i++) {
expandedR[i] = R[E[i]-1];
}
int temp[] = xor(expandedR, roundKey);
int output[] = sBlock(temp);
return output;
}
private static int[] xor(int[] a, int[] b) {
int answer[] = new int[a.length];
for(int i=0 ; i < a.length ; i++) {
```

```
answer[i] = a[i]^b[i];
}
return answer;
}
private static int[] sBlock(int[] bits) {
int output[] = new int[32];
for(int i=0 ; i < 8 ; i++) {
int row[] = new int [2];
row[0] = bits[6*i];
row[1] = bits[(6*i)+5];
String sRow = row[0] + "" + row[1];
int column[] = new int[4];
column[0] = bits[(6*i)+1];
column[1] = bits[(6*i)+2];
column[2] = bits[(6*i)+3];
column[3] = bits[(6*i)+4];
String sColumn = column[0] +""+ column[1] +""+
column[2] +""+ column[3];
int iRow = Integer.parseInt(sRow, 2);
int iColumn = Integer.parseInt(sColumn, 2);
int x = S[i][(iRow*16) + iColumn];
String s = Integer.toBinaryString(x);
```

```
while(s.length() < 4) {
s = "0" + s;
}
for(int j=0 ; j < 4 ; j++) {
output[(i*4) + j] = Integer.parseInt(s.charAt(j) + "");
}
}
int finalOutput[] = new int[32];
for(int i=0 ; i < 32 ; i++) {
finalOutput[i] = output[P[i]-1];
}
return finalOutput;
}
private static int[] leftShift(int[] bits, int n) {
int answer[] = new int[bits.length];
System.arraycopy(bits, 0, answer, 0, bits.length);
for(int i=0 ; i < n ; i++) {
int temp = answer[0];
for(int j=0 ; j < bits.length-1 ; j++) {
answer[j] = answer[j+1];
}
answer[bits.length-1] = temp;}
```

```
return answer;
}
private static void displayBits(int[] bits) {
for(int i=0 ; i < bits.length ; i+=4) {
String output = new String();
for(int j=0 ; j < 4 ; j++) {
output += bits[i+j]; }
System.out.print(Integer.toHexString(Integer.parseInt(output, 2)));}
System.out.println(); }}
```

OUTPUT

```
Round 13:
Round key = 75c01002c58d
L = 9bfc7fef
R = 51f95b37

-------------
Round 14:
Round key = 4004d35a81c5
L = 51f95b37
R = 270259eb

-------------
Round 15:
Round key = 21b5000706c5
L = 270259eb
R = 428921da

-------------
Round 16:
Round key = 2c1224b52308
L = 428921da
R = 25ffc680
Decrypted text: 7896543212589637
BUILD SUCCESSFUL (total time: 44 seconds)
```

RSA

AIM :

To implement RSA Encryption Algorithm Using C/Java.

ALGORITHM :

1. Generate two large random primes, p and q, of approximately equal size such that their product n = pq is of the required bit length, e.g. 1024 bits.
2. Compute n = pq and (phi) = (p-1)(q-1).
3. Choose an integer e, 1 <e < phi, such that gcd(e, phi) = 1.
4. Compute the secret exponent d, 1 <d< phi, such that ed = 1 (mod phi).
5. The public key is (n, e) and the private key (d, p, q). Keep all the values d, p, q and phi secret.
6. Computes the ciphertext c = m^2 mod n. where m is plain text.

7. Compute m = cd mod n and Extract the plaintext from the message representative m.

RESULT :

THE OUTPUT HAS BEEN OBTAINED SUCESSFULLY.

PROGRAM

```
import java.io.DataInputStream;
import java.io.IOException;
import java.math.BigInteger;
import java.util.Random;
public class Rsa
{
private BigInteger P;
private BigInteger Q;
private BigInteger N;
private BigInteger PHI;
private BigInteger e;
private BigInteger d;
```

```
private int maxLength = 1024;
private Random R;
public Rsa()
{
R = new Random();
P = BigInteger.probablePrime(maxLength, R);
Q = BigInteger.probablePrime(maxLength, R);
N = P.multiply(Q);
PHI=P.subtract(BigInteger.ONE).multiply(
Q.subtract(BigInteger.ONE));
e = BigInteger.probablePrime(maxLength / 2, R);
while (PHI.gcd(e).compareTo(BigInteger.ONE) > 0 &&
e.compareTo(PHI) < 0)
{
e.add(BigInteger.ONE);
}
d = e.modInverse(PHI);
}
public Rsa(BigInteger e, BigInteger d, BigInteger N)
{
this.e = e;
```

```
this.d = d;
this.N = N;
}
public static void main (String [] arguments) throws IOException
{
Rsa rsa = new Rsa();
DataInputStream input = new DataInputStream(System.in);
String inputString;
System.out.println("Enter message you wish to send.");
inputString = input.readLine();
System.out.println("Encrypting the message: " + inputString);
System.out.println("The message in bytes is:: "
+ bToS(inputString.getBytes()));
byte[] cipher = rsa.encryptMessage(inputString.getBytes());
byte[] plain = rsa.decryptMessage(cipher);
System.out.println("Decrypting Bytes: " + bToS(plain));
System.out.println("Plain message is: " + new String(plain));
}
private static String bToS(byte[] cipher)
{
```

```
String temp = "";
for (byte b : cipher)
{temp += Byte.toString(b);
}return temp;}
public byte[] encryptMessage(byte[] message){
Return(newBigInteger(message)).modPow(e,
N).toByteArray();
}public byte[] decryptMessage(byte[] message)
{return(newBigInteger(message)).modPow(d,
N).toByteArray(); }}
```

OUTPUT

```
debug:
Enter message you wish to send.
hi how are you
Encrypting the message: hi how are you
The message in bytes is:: 104105321041111193297114101321211111117
Decrypting Bytes: 104105321041111193297114101321211111117
Plain message is: hi how are you
BUILD SUCCESSFUL (total time: 13 seconds)
```

HASH FUNCTIONS

AIM :

To implement the Hash Function Using C/Java.

ALGORITHM :

1. Import the java.security.MessageDigest class for generating MD5 hash.
2. Declare a String variable to hold the string that needs to be hashed.
3. Convert the string into a byte array using the getBytes() method of String class.
4. Create an instance of the MessageDigest class using the getInstance() method, passing "MD5" as a parameter.
5. Update the MessageDigest object with the byte array using the update() method.
6. Generate the hash using the digest() method, which returns an array of bytes representing the hash.

7. Convert the byte array into a hexadecimal string using the String.format() method.

RESULT :

THE OUTPUT HAS BEEN OBTAINED SUCESSFULLY.

PROGRAM

```
import java.io.*;
import java.math.BigInteger;
import java.lang.*;
public class Hashh {
static String md5_constant[] = new String[64];
public static void md5_key_generation() {
try {
double k[] = new double[64];
String key[] = new String[64];
for (int i = 0; i <= 63; i++) {
k[i] = Math.floor(Math.abs(Math.sin(i + 1)) * Math.pow(2, 32));
md5_constant[i] = Long.toBinaryString(Double.doubleToRawLongBits(k[i]));
md5_constant[i] = md5_constant[i] + '0';
```

```
}
} catch (Exception e) {
System.out.println(e);
}}
public static String hash(String input) {
String hashcode = "";
String temp = "";
String[] x = new String[64];
String a = "00000001001000110100010101100111";
String b = "10001001101010111100110111101111";
String c = "11111110110111001011101010011000";
String d = "01110110010101000011001000010000";
int t = 0;
for (int i = 0; i < 16; i++) {
x[i] = "";
for (int j = 0; j < 32; j++) {
if (t < input.length()) {
x[i] = x[i] + input.charAt(t);
} else {
x[i] = x[i] + '0';
}
t++;
}}
```

```
int round = 1;
int ccount = 0;
while (round <= 4) {
for (int r = 0; r < 16; r++) {
if (round == 1) {
temp = gf(b, c, d);
} else if (round == 2) {
temp = gg(b, c, d);
} else if (round == 3) {
temp = gh(b, c, d);
} else if (round == 4) {
temp = gi(b, c, d);
}
temp = xor(a, temp);
if (round == 1) {
temp = xor(temp, x[r]);
} else if (round == 2) {
temp = xor(temp, x[(1 + (5 * r)) % 16]);
} else if (round == 3) {
temp = xor(temp, x[(5 + (3 * r)) % 16]);
} else if (round == 4) {
temp = xor(temp, x[(7 * r) % 16]);
}
```

```
temp = xor(temp, md5_constant[ccount]);
temp = temp.substring(24, 32) + temp;
temp = temp.substring(0, 32);
temp = xor(temp, b);
a = d;
d = c;
c = b;
b = temp;
ccount++;
}
round++;
}
hashcode = a + b + c + d;
BigInteger bi = new BigInteger(hashcode, 2);
hashcode = bi.toString(16);
return hashcode;
}
public static String gf(String b, String c, String d) {
String temp = "";
temp = xor(b, c);
temp = xor(temp, d);
return temp;
}
```

```
public static String gg(String b, String c, String d) {
String temp = "";
temp = xor(b, c);
temp = xor(temp, d);
return temp;
}
public static String gh(String b, String c, String d) {
String temp = "";
temp = xor(b, c);
temp = xor(temp, d);
return temp;
}
public static String gi(String b, String c, String d) {
String temp = "";
temp = xor(b, c);
temp = xor(temp, d);
return temp;
}
public static String xor(String a, String b) {
String temp = "";
for (int i = 0; i < 32; i++) {
if (a.charAt(i) == b.charAt(i)) {
temp = temp + '0';
```

```
} else {
temp = temp + '1';
}
}
return temp;
}
public static void main(String[] args) throws IOException {
md5_key_generation();
BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
System.out.println("Enter the string to be hashed");
String input = br.readLine();
String hashcode = hash(input);
System.out.println("The hashcode is " + hashcode);
}}
```

OUTPUT

Enter The Input String: hello

Hash code: 2UeqD7PDTPDTE

BLOWFISH

AIM :

To implement the Blowfish Algorithm Logic Using C/Java.

ALGORITHM :

1. Import the necessary classes: javax.crypto.Cipher, javax.crypto.spec.SecretKeySpec, and javax.crypto.KeyGenerator.
2. Generate a secret key using the KeyGenerator class, setting the key size to the desired value (in bits).
3. Initialize the Cipher object with the "Blowfish" algorithm and the secret key using the SecretKeySpec class.
4. Convert the plaintext message into a byte array.

5. Encrypt the message using the Cipher object's doFinal() method, which returns a byte array representing the encrypted message.
6. To decrypt the message, initialize the Cipher object with the "Blowfish" algorithm and the same secret key using the SecretKeySpec class.
7. Convert the encrypted message byte array back into a plaintext message byte array using the Cipher object's "doFinal()" method with the Cipher.DECRYPT_MODE parameter.

RESULT :

THE OUTPUT HAS BEEN OBTAINED SUCESSFULLY.

PROGRAM

```
N import javax.crypto.Cipher;

import javax.crypto.KeyGenerator;
```

```
import javax.crypto.SecretKey;
import javax.swing.JOptionPane;
public class Blow
{
public static void main(String[] args)throws Exception
{
KeyGeneratorkeygenerator=KeyGenerator.getInstance("Blowfish");
Cipher cipher = Cipher.getInstance("Blowfish");
SecretKey secretkey = keygenerator.generateKey();
cipher.init(Cipher.ENCRYPT_MODE, secretkey);
String inputText = JOptionPane.showInputDialog("Input your message:");
byte[] encrypted = cipher.doFinal(inputText.getBytes());
cipher.init(Cipher.DECRYPT_MODE, secretkey);
byte[] decrypted = cipher.doFinal(encrypted);
JOptionPane.showMessageDialog(JOptionPane.getRootFrame(),
```

```
"\nEncrypted text: " + new String(encrypted) + "\n" +
"\nDecrypted text: " + new String(decrypted));
System.exit(0);
}}
```

OUTPUT

DIFFIE-HELLMAN

AIM :

To implement the Diffie-Hellman Key Exchange mechanism Using C/Java.

ALGORITHM :

1. GLOBAL PUBLIC ELEMENTS,Select any prime no: 'q',Calculate the primitive root of q: 'a' such that a<q.
2. ASYMMETRIC KEY GENERATION BY USER 'A' Select a random number as the private key X, where X_1< q Calculate the public key Y, where Y_1 = a mod q.
3. KEY GENERATION BY USER 'B',Select a random number as the private key X, where XB< q Calculate the public key Y, where Y_1 = a* mod q.
4. Exchange the values of public key between A & B.
5. SYMMETRIC KEY (K) GENERATION BY USER 'A',K=YA mod q.

6. SYMMETRIC KEY (K) GENERATION BY USER 'B',K= YAB mod q.

RESULT :

THE OUTPUT HAS BEEN OBTAINED SUCESSFULLY.

PROGRAM

```
package Diffi;

import java.util.*;

import java.math.BigInteger;

public class Diffi

{

final static BigInteger one = new BigInteger("1");

public static void main(String args[]) {

Scanner stdin = new Scanner(System.in);

BigInteger p;
```

```
System.out.println("Enter the approximate value of p you want.");

String ans = stdin.next();

p = getNextPrime(ans);

System.out.println("Your prime is "+p+".");

System.out.println("Now, enter a number in between 2 and p-1.");

BigInteger g = new BigInteger(stdin.next());

System.out.println("Person A: enter your secret number now.");

BigInteger a = new BigInteger(stdin.next());

BigInteger resulta = g.modPow(a,p);

System.out.println("Person A sends to person B "+resulta+".");

System.out.println("Person B: enter your secret number now.");

BigInteger b = new BigInteger(stdin.next());

BigInteger resultb = g.modPow(b,p);
```

```
System.out.println("Person B sends to person A "+resultb+".");

BigInteger KeyACalculates = resultb.modPow(a,p);

BigInteger KeyBCalculates = resulta.modPow(b,p);

System.out.println("A takes "+resultb+" raises it to the power "+a+" mod "+p);

System.out.println("TheKeyAcalculatesis"+KeyACalculates+".");

System.out.println("B takes "+resulta+" raises it to the power "+b+" mod "+p);

System.out.println("TheKeyBcalculatesis"+KeyBCalculates+".");

}

public static BigInteger getNextPrime(String ans) {

BigInteger test = new BigInteger(ans);

while (!test.isProbablePrime(99))

test = test.add(one);

return test;}}
```

OUTPUT

```
debug:
Enter the approximate value of p you want.
10
Your prime is 11.
Now, enter a number in between 2 and p-1.
5
Person A: enter your secret number now.
125698
Person A sends to person B 4.
Person B: enter your secret number now.
542365
Person B sends to person A 1.
A takes 1 raises it to the power 125698 mod 11
The Key A calculates is 1.
B takes 4 raises it to the power 542365 mod 11
The Key B calculates is 1.
BUILD SUCCESSFUL (total time: 2 minutes 15 seconds)
```

RC4

AIM :

To implement the RC4 logic in Java Using Java cryptography; encrypt the text Hello world using Blowfish. Create your own key using Java key tool.

ALGORITHM :

1. Import the necessary classes from the Java Cryptography API.

   import javax.crypto.Cipher;

   import javax.crypto.KeyGenerator;

   import javax.crypto.SecretKey;
2. Create an instance of the KeyGenerator class to generate a Blowfish key.

   KeyGenerator keyGenerator = KeyGenerator.getInstance("Blowfish");
3. Set the key size of the key generator (in bits).

   keyGenerator.init(128);

4. Generate a secret key using the generateKey() method of the KeyGenerator class.

   SecretKey secretKey = keyGenerator.generateKey();

5. Create an instance of the Cipher class for encryption and initialize it with the secret key.

   Cipher cipher = Cipher.getInstance("Blowfish");
   cipher.init(Cipher.ENCRYPT_MODE, secretKey);

6. Convert the plaintext string "Hello world" to a byte array.

   byte[] plaintext = "Hello world".getBytes();

7. Encrypt the plaintext using the doFinal() method of the Cipher class.

   byte[] ciphertext = cipher.doFinal(plaintext);

8. Convert the ciphertext byte array to a string.

   String ciphertextString = new String(ciphertext);

9. Print the ciphertext string to the console.

   System.out.println("Ciphertext: " + ciphertextString);

10. Save the secret key to a file using the write() method of the SecretKey class.

```
Files.write(Paths.get("secretKey.txt"),
secretKey.getEncoded());
```

RESULT :

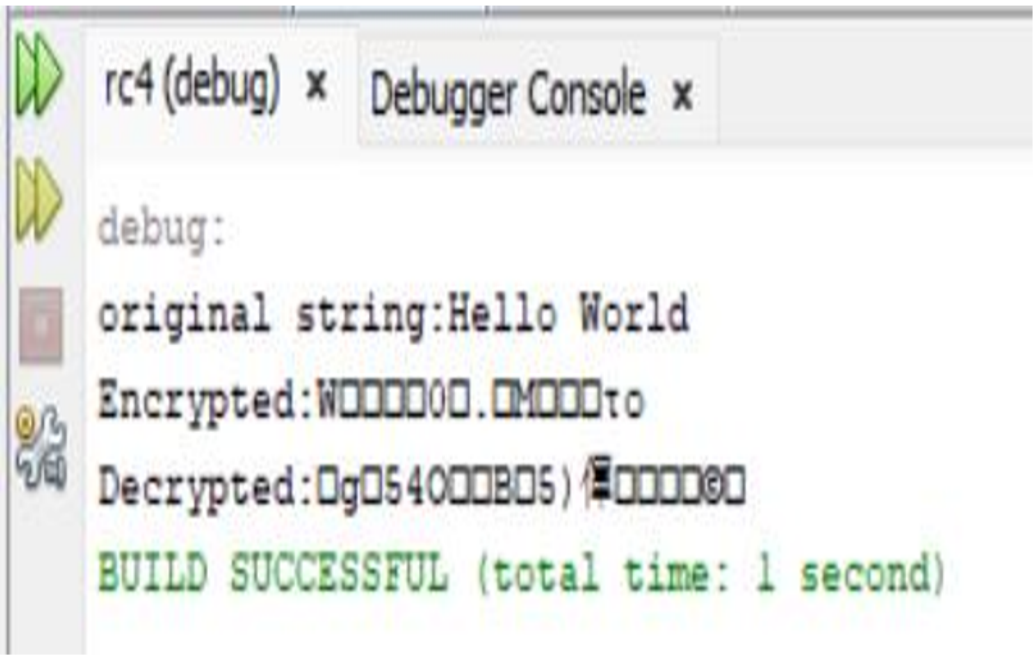
THE OUTPUT HAS BEEN OBTAINED SUCESSFULLY.

PROGRAM

```
import java.security.Key;
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.swing.JOptionPane;
public class Rc4
{public static void main(String[] args) throws Exception{
 KeyGenerator keyGenerator =
KeyGenerator.getInstance("Blowfish");
```

```
 SecretKey secretKey = keyGenerator.generateKey();
 Cipher cipher = Cipher.getInstance("Blowfish");
 cipher.init(Cipher.ENCRYPT_MODE,secretKey);
String inputText = "Hello World";
byte[] encrypted=cipher.doFinal(inputText.getBytes());
cipher.init(Cipher.ENCRYPT_MODE,secretKey);
byte[] decrypted=cipher.doFinal(encrypted);
System.out.println("original string:"+inputText);
System.out.println("Encrypted:"+new String(encrypted));
System.out.println("Decrypted:"+new String(decrypted));}}
```

OUTPUT



DIGITAL SIGNATURE

AIM :

To Implement the SIGNATURE SCHEME -Digital Signature Standard

ALGORITHM :

1. The first part of the DSA algorithm is the public key and private key generation through some steps, which can be told as:

   •Firstly, choose a prime number q, which is called the prime divisor in this.

   • Then, choose another primer number p, such that p - 1 mod q = 0 p is called the prime modulus in this.

   •Then, choose an integer g, such that 1 < g < p ,g^ * q mod p = 1 and g=h^ * ((p - 1) / q) mod pq is also called g's multiplicative order modulo p in this algorithm.

• Then, choose an integer, such that 0 < x < q for this.

•Now, compute y as g ^ ** x mod p.

•Thus, Package the public key as {p,q,g,y) is this

• And, Package the private key as {p,q,g,x} is this.

2. Then, the second part of the DSA algorithm is the signature generation and signature verification in this algorithm, which can be told as:

• Firstly, generate the message digest h, using a hash algorithm like SHA1.

•Then, generate a random number k, such that 0 < k < q

• Then, Computer as (g ^ ** k mod p) mod q. If r = 0 , select a different k.

•And, Compute i, such that k ^ * i mod q = 1 i is called the modular multiplicative inverse of k modulo q in this.

•Then, Computes s =i^ * (h+r^ * x) mod q. If s = 0 select a different k.Thus, Package the digital signature as {r,s).

3. Then, to verify a message signature, the receiver of the message and the digital signature can follow these further steps as:

•Firstly, Generate the message digest h, using the same hash algorithm.

•Then, Compute w, such that s*w mod q = 1 w is called the modular multiplicative inverse of s modulo q in this.

•Then, Compute u1=h^ * wmoda .

4. And, Compute u2=r^ * wmodc . • Then, Compute v =(((g^ * u1)^ * (y^ * u2)) mod p) mod q.
5. Wherever, If v ==r , the digital signature is valid.

RESULT :

THE OUTPUT HAS BEEN OBTAINED SUCESSFULLY.

PROGRAM

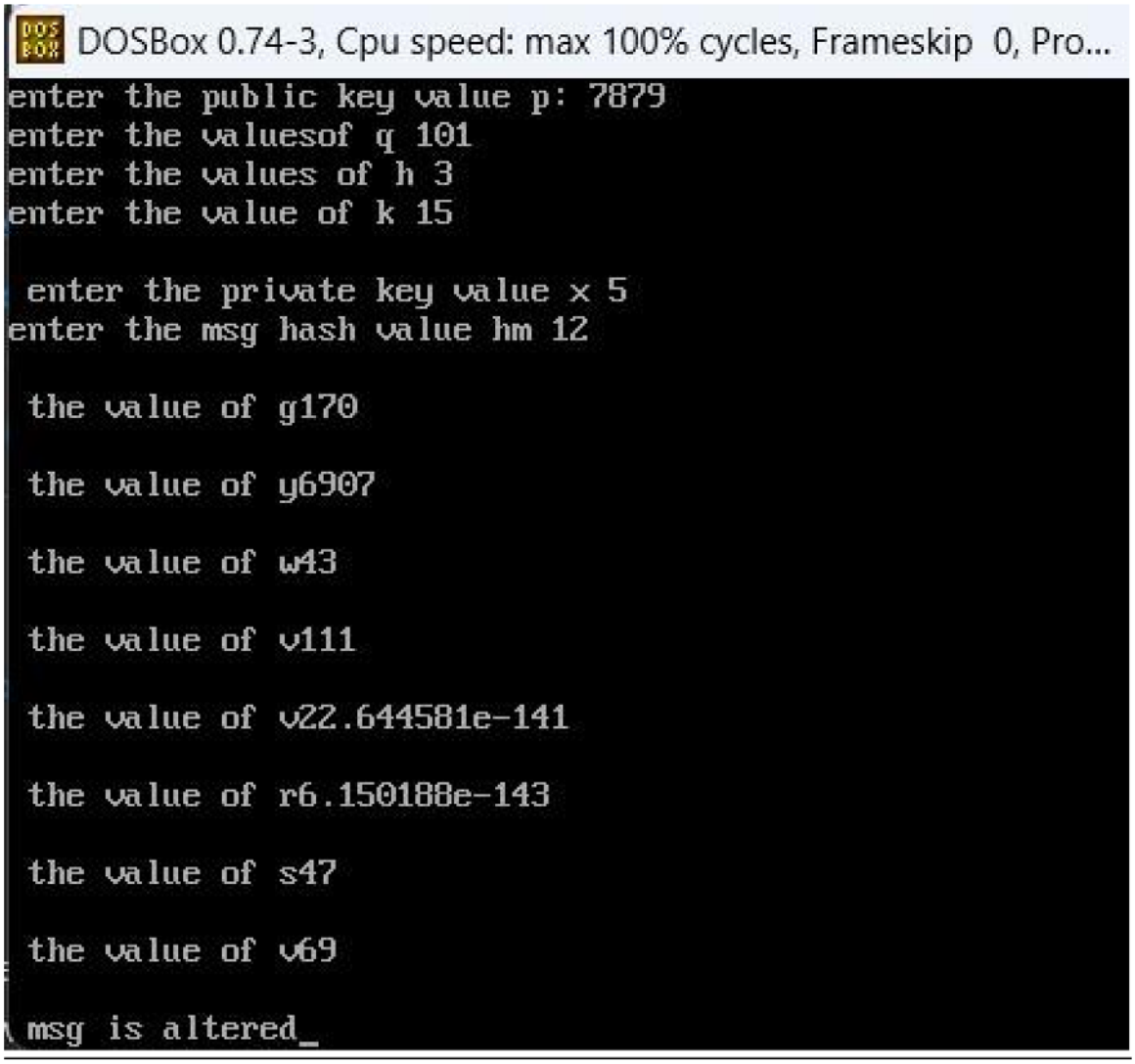
```
#include<stdio.h>
#include<conio.h>
#include<iostream.h>
#include<math.h>
#include<string.h>
class dss
{       double p,m,g,q,y,x,s,r,v,w,v1,v2,h,k,hm;
        public:
       double inverse(double e,double d)
```

```
        { double t;
  int x=e,y=d,t1;
  t1=1;
          while((x*t1)%y!=1)
  t1++;
          t=t1;
  return t;
}
void keygen()
{       double t,t1;
cout<<"enter the public key value p:" ;
cin>>p;
cout<<"enter the valuesof q";
cin>>q;
cout<<"enter the values of h";
cin>>h;
cout<<"enter the value of k";
cin>>k;
cout<<"\n enter the private key value x";
cin>>x;
cout<<"enter the msg hash value hm";
cin>>hm;
```

```
t=(p-1)/q;
double a=1;
for(int i=0;i<t;i++)
a=fmod(h*a,p);
g=a;
double b=1;
for(i=0;i<k;i++)
b=fmod(g*b,p);
y=b;
t1=(hm+(x*t))*inverse(k,q);
s=fmod(t1,q);
w=inverse(s,q);
v1=fmod(hm*w,q);
v2=fmod(r*w,q);
a=1;
for(i=0;i<v1;i++)
a=fmod(g*a,p);
b=1;
for(i=0;i<v2;i++)
a=fmod(g*b,p);
v=fmod(fmod(a*b,p),q);
cout<<"\n the value of g"<<g<<endl;
```

```
cout<<"\n the value of y"<<y<<endl;
cout<<"\n the value of w"<<w<<endl;
cout<<"\n the value of v1"<<v1<<endl;
cout<<"\n the value of v2"<<v2<<endl;
cout<<"\n the value of r"<<r<<endl;
cout<<"\n the value of s"<<s<<endl;
cout<<"\n the value of v"<<v<<endl;
if(v==r)
cout<<"\n the msg is not altered";
else
cout<<"\n msg is altered";
}
};
void main()
{       clrscr();
dss c;
c.keygen();
getch();
}
```

OUTPUT


DOSBox 0.74-3, Cpu speed: max 100% cycles, Frameskip 0, Pro...
enter the public key value p: 7879
enter the valuesof q 101
enter the values of h 3
enter the value of k 15
enter the private key value x 5
enter the msg hash value hm 12
the value of g170
the value of y6907
the value of w43
the value of v111
the value of v22.644581e-141
the value of r6.150188e-143
the value of s47
the value of v69
msg is altered_