Program Statement 1: 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.

```
#include <stdio.h>

int main()
{
   char str[] = "Hello World";
   printf("Original string: %s\n\n", str);

   printf("Bitwise AND operation: ");
   for(int i=0;str[i]!='\0';i++){
      str[i] = str[i] & 127;
      printf("%c",str[i]);
   }
   printf("\n\n");

   printf("Bitwise OR operation: ");
   for(int i=0;str[i]!='\0';i++){
      str[i] = str[i] | 127;
      printf("%c",str[i]);
   }
   printf("\n\n");

   printf("Bitwise XOR operation: ");
   for(int i=0;str[i]!='\0';i++){
      str[i] = str[i] ^ 127;
      printf("%c",str[i]);
   }
   printf("\n\n");

   return 0;
}
```

OUTPUT:

```
Original string: Hello World
Bitwise AND operation: Hello World
Bitwise OR operation:
Bitwise XOR operation:
```

Program Statement 2: Write a Java program to perform encryption and decryption using the following algorithms:

a. Ceasar cipher b. Playfair cipher

CEASAR CIPHER:

```
import java.util.*;

class ccipher{

 static String encrypt(String s,int key){
   String ans="";
   for(int i=0;i<s.length();i++){
      char c=s.charAt(i),add='.';
      if(Character.isUpperCase(c))
       add=(char)('A'+(c-'A' +key)%26);

       else if(Character.isLowerCase(c))
       add=(char)('a'+(c-'a' +key)%26);
       else
       add=c;
      ans+=add;
      }
      return ans;

}


static String decrypt(String s,int key){
 return encrypt(s,26-key);
}

 public static void main(String args[]){

Scanner sc=new Scanner(System.in);
System.out.println("Enter text to encrypt: ");
String plainText=sc.nextLine();
System.out.println("Enter shift value: ");
int shift=sc.nextInt();
System.out.println("Original text is "+plainText);
String encrypted=encrypt(plainText,shift);
System.out.println("Encrypted Text is "+encrypted);
System.out.println("Decrypted Text is "+decrypt(encrypted,shift));
}
}
```

OUTPUT:

```
Enter text to encrypt:
COMPUTER SCIENCE
Enter shift value:
3
Original text is COMPUTER SCIENCE
Encrypted Text is FRPSXWHU VFLHQFH
Decrypted Text is COMPUTER SCIENCE
```

b) **PLAYFAIR CIPHER:**

```
import java.awt.Point;
import java.util.Scanner;

public class PlayfairCipher
{
   private int length = 0;
   private String [][] table;
   public static void main(String args[])
{
   PlayfairCipher pf = new PlayfairCipher();
}
private PlayfairCipher()
{
   System.out.print("Enter the key for playfair cipher: ");
   Scanner sc = new Scanner(System.in);
   String key = parseString(sc);
   while(key.equals(""))
   key = parseString(sc);
   table = this.cipherTable(key);
   System.out.print("Enter the plaintext to be encipher: ");
   String input = parseString(sc);
   while(input.equals(""))
   input = parseString(sc);
   String output = cipher(input);
   String decodedOutput = decode(output);
   this.keyTable(table);
   this.printResults(output,decodedOutput);
}
private String parseString(Scanner sc)
{
   String parse = sc.nextLine();
   parse = parse.toUpperCase();
   parse = parse.replaceAll("[^A-Z]", "");
   parse = parse.replace("J", "I");
   return parse;
}
private String[][] cipherTable(String key)
{
   String[][] playfairTable = new String[5][5];
```

```
    String keyString = key + "ABCDEFGHIKLMNOPQRSTUVWXYZ";
    for(int i = 0; i < 5; i++)
    for(int j = 0; j < 5; j++)
    playfairTable[i][j] = "";
    for(int k = 0; k < keyString.length(); k++)
    {
        boolean repeat = false;
        boolean used = false;
        for(int i = 0; i < 5; i++)
        {
            for(int j = 0; j < 5; j++)
            {
                if(playfairTable[i][j].equals("" + keyString.charAt(k)))
                {
                    repeat = true;
                }
                else if(playfairTable[i][j].equals("") && !repeat && !used)
                {
                    playfairTable[i][j] = "" + keyString.charAt(k);
used = true;
                }
            }
        }
    }
    return playfairTable;
}
private String cipher(String in)
{
    length = (int) in.length() / 2 + in.length() % 2;
    for(int i = 0; i < (length - 1); i++)
    {
        if(in.charAt(2 * i) == in.charAt(2 * i + 1))
        {
            in = new StringBuffer(in).insert(2 * i + 1, 'X').toString();
            length = (int) in.length() / 2 + in.length() % 2;
        }
    }
    String[] digraph = new String[length];
    for(int j = 0; j < length ; j++)
    {
        if(j == (length - 1) && in.length() / 2 == (length - 1))
        in = in + "X";
        digraph[j] = in.charAt(2 * j) +""+ in.charAt(2 * j + 1);
    }
    String out = "";
    String[] encDigraphs = new String[length];
    encDigraphs = encodeDigraph(digraph);
    for(int k = 0; k < length; k++)
    out = out + encDigraphs[k];
    return out;
}

private String[] encodeDigraph(String di[])
{
    String[] encipher = new String[length];
    for(int i = 0; i < length; i++)
    {
        char a = di[i].charAt(0);
        char b = di[i].charAt(1);
        int r1 = (int) getPoint(a).getX();
        int r2 = (int) getPoint(b).getX();
        int c1 = (int) getPoint(a).getY();
        int c2 = (int) getPoint(b).getY();
        if(r1 == r2)
        {
            c1 = (c1 + 1) % 5;
            c2 = (c2 + 1) % 5;
        }
        else if(c1 == c2)
        {
            r1 = (r1 + 1) % 5;
            r2 = (r2 + 1) % 5;
        }
        else
        {
            int temp = c1;
            c1 = c2;
            c2 = temp;
        }
        encipher[i] = table[r1][c1] + "" + table[r2][c2];
```

```
   }
   return encipher;
}

private String decode(String out)
{
   String decoded = "";
   for(int i = 0; i < out.length() / 2; i++)
   {
      char a = out.charAt(2*i);
      char b = out.charAt(2*i+1);
      int r1 = (int) getPoint(a).getX();
      int r2 = (int) getPoint(b).getX();
      int c1 = (int) getPoint(a).getY();
      int c2 = (int) getPoint(b).getY();
      if(r1 == r2)
      {
         c1 = (c1 + 4) % 5;
         c2 = (c2 + 4) % 5;
      }
      else if(c1 == c2)
      {
         r1 = (r1 + 4) % 5;
         r2 = (r2 + 4) % 5;
      }
      else
      {
         int temp = c1;
         c1 = c2;
         c2 = temp;
      }
      decoded = decoded + table[r1][c1] + table[r2][c2];
   }
   return decoded;
}

private Point getPoint(char c)
{
   Point pt = new Point(0,0);
   for(int i = 0; i < 5; i++)
   for(int j = 0; j < 5; j++)
   if(c == table[i][j].charAt(0))
   pt = new Point(i,j);
   return pt;
}

private void keyTable(String[][] printTable)
{
   System.out.println("Playfair Cipher Key Matrix: ");
   System.out.println();
   for(int i = 0; i < 5; i++)
   {
      for(int j = 0; j < 5; j++)
      {
         System.out.print(printTable[i][j]+" ");
      }
      System.out.println();
   }
   System.out.println();
}

private void printResults(String encipher, String dec)
{
   System.out.print("Encrypted Message: ");
   System.out.println(encipher);
   System.out.println();
   System.out.print("Decrypted Message: ");
   System.out.println(dec);
}

}
```

OUTPUT:

```
Enter the key for playfair cipher: MONARCHY

Enter the plaintext to be encipher: NITTE MEENAKSHI INSTITUTE OF TECHNOLOGY

Playfair Cipher Key Matrix:

M O N A R

C H Y B D

E F G I K
```

```
L P Q S T

U V W X Z

Encrypted Message: AGSZLKCLGMRIPBSAGATLKSZLFMKPLEYOMPNFBW

Decrypted Message: NITXTEMEENAKSHIXINSTITUTEOFTECHNOLOGYX
```

Program 3: Write a C program to implement the following:

a. Vigenere Cipher using a Vigenere table.

b. Rail fence Cipher using row and column transformation

a) **VIGENERE CIPHER:**

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

void printVigenereTable()
{
   printf("Vigenere Table ");
   printf("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 \n");
   for(int i = 0; i < 26; i++){
      printf("%c", 'A' + i);
      for(int j = 0; j < 26; j++){
         printf("%c", 'A' + (i + j) % 26);
      }
      printf("\n");
   }
}

void encrypt()
{
   char plaintext[128];
   char key[16];
   printf("Enter the plain text: ");
   scanf(" %[^\n]", plaintext);
   getchar();
   printf("Enter the key: ");
   scanf(" %[^\n]", key);
   getchar();

   printf("Cipher text is: ");
   for(int i = 0, j = 0; i < strlen(plaintext); i++, j++){
      if(j >= strlen(key)){
         j = 0;
      }
      int shift = toupper(key[j]) - 'A';
```

```
        char encryptChar = ((toupper(plaintext[i]) - 'A' + shift) % 26) + 'A';
        printf("%c", encryptChar);
    }
    printf("\n");
}

void decrypt()
{
    char ciphertext[128];
    char key[16];
    printf("Enter the chipher text; ");
    scanf(" %[^\n]", ciphertext);
    getchar();
    printf("Enter the key: ");
    scanf(" %[^\n]", key);
    getchar();

    printf("decrypted text: ");
    for(int i=0, j=0; i < strlen(ciphertext); i++, j++){
        if(j >= strlen(key)){
            j = 0;
        }
        int shift = toupper(key[j]) - 'A';
        char decryptChar = ((toupper(ciphertext[i]) - 'A' - shift + 26) % 26) + 'A';
        printf("%c", decryptChar);
    }
    printf("\n");
}

int main() {
    int option;
    while (1) {
        printf("\n1. Encrypt");
        printf("\n2. Decrypt");
        printf("\n3. Print Vigenère Table");
        printf("\n4. Exit\n");
        printf("\nEnter your option: ");
        scanf("%d", &option);

        switch (option) {
            case 1:
                encrypt();
                break;
            case 2:
                decrypt();
                break;
            case 3:
                printVigenereTable();
                break;
            case 4:
                exit(0);
            default:
                printf("\nInvalid selection! Try again.\n");
                break;
        }
    }
    return 0;
}
```

OUTPUT:

```
1. Encrypt
2. Decrypt
3. Print Vigenère Table
4. Exit

Enter your option: 1
Enter the plain text: EXPLAINATION
Enter the key: leg
Cipher text is: PBVWEOYEZTST

1. Encrypt
2. Decrypt
3. Print Vigenère Table
4. Exit

Enter your option: 2
Enter the chipher text; PBVWEOYEZTST
Enter the key: leg
decrypted text: EXPLAINATION

1. Encrypt
```

```
2. Decrypt
3. Print Vigenère Table
4. Exit

Enter your option: 3
Vigenere Table 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
AABCDEFGHIJKLMNOPQRSTUVWXYZ
BBCDEFGHIJKLMNOPQRSTUVWXYZA
CCDEFGHIJKLMNOPQRSTUVWXYZAB
DDEFGHIJKLMNOPQRSTUVWXYZABC
EEFGHIJKLMNOPQRSTUVWXYZABCD
FFGHIJKLMNOPQRSTUVWXYZABCDE
GGHIJKLMNOPQRSTUVWXYZABCDEF
HHIJKLMNOPQRSTUVWXYZABCDEFG
IIJKLMNOPQRSTUVWXYZABCDEFGH
JJKLMNOPQRSTUVWXYZABCDEFGHI
KKLMNOPQRSTUVWXYZABCDEFGHIJ
LLMNOPQRSTUVWXYZABCDEFGHIJK
MMNOPQRSTUVWXYZABCDEFGHIJKL
NNOPQRSTUVWXYZABCDEFGHIJKLM
OOPQRSTUVWXYZABCDEFGHIJKLMN
PPQRSTUVWXYZABCDEFGHIJKLMNO
QQRSTUVWXYZABCDEFGHIJKLMNOP
RRSTUVWXYZABCDEFGHIJKLMNOPQ
SSTUVWXYZABCDEFGHIJKLMNOPQR
TTUVWXYZABCDEFGHIJKLMNOPQRS
UUVWXYZABCDEFGHIJKLMNOPQRST
VVWXYZABCDEFGHIJKLMNOPQRSTU
WWXYZABCDEFGHIJKLMNOPQRSTUV
XXYZABCDEFGHIJKLMNOPQRSTUVW
YYZABCDEFGHIJKLMNOPQRSTUVWX
ZZABCDEFGHIJKLMNOPQRSTUVWXY

1. Encrypt
2. Decrypt
3. Print Vigenere Table
4. Exit

Enter your option: 4
```

b) **RAILFENCE CIPHER:**

```
#include<stdio.h>

#include<string.h>

#include<stdlib.h>


void encryptMessage(char *str, int rails);

void decryptMessage(char *str, int rails);


void encryptMessage(char *str, int rails) {

    int i, j, len, count, code[100][1000];

    len = strlen(str);


    for(i = 0; i < rails; i++) {

        for(j = 0; j < len; j++) {

            code[i][j] = 0;

        }

    }


    count = 0;

    j = 0;

    while(j < len) {

        if(count % 2 == 0) {

            for(i = 0; i < rails; i++) {

                if(j < len)

                    code[i][j] = (int)str[j];

                j++;

            }

        } else {

            for(i = rails - 2; i > 0; i--) {

                if(j < len)
```

```
               code[i][j] = (int)str[j];
            j++;
         }
      }
      count++;
   }

   printf("Rail Fence Pattern:\n");
   for(i = 0; i < rails; i++) {
      for(j = 0; j < len; j++) {
         if(code[i][j] != 0)
            printf("%c ", code[i][j]);
         else
            printf("  ");
      }
      printf("\n");
   }

   printf("Encrypted Message: ");
   for(i = 0; i < rails; i++) {
      for(j = 0; j < len; j++) {
         if(code[i][j] != 0)
            printf("%c", (char)code[i][j]);
      }
   }
   printf("\n");
}

void decryptMessage(char *str, int rails) {
   int i, j, len, count, k, code[100][1000];
```

```
len = strlen(str);

for(i = 0; i < rails; i++) {
    for(j = 0; j < len; j++) {
        code[i][j] = 0;
    }
}

count = 0;
j = 0;
while(j < len) {
    if(count % 2 == 0) {
        for(i = 0; i < rails; i++) {
            if(j < len)
                code[i][j] = 1;
            j++;
        }
    } else {
        for(i = rails - 2; i > 0; i--) {
            if(j < len)
                code[i][j] = 1;
            j++;
        }
    }
    count++;
}

k = 0;
for(i = 0; i < rails; i++) {
    for(j = 0; j < len; j++) {
            if(code[i][j] == 1) {
                code[i][j] = (int)str[k];
                k++;
            }
        }
    }

    printf("Decrypted Message: ");
    count = 0;
    j = 0;
    while(j < len) {
        if(count % 2 == 0) {
            for(i = 0; i < rails; i++) {
                if(code[i][j] != 0) {
                    printf("%c", (char)code[i][j]);
                    j++;
                }
            }
        } else {
            for(i = rails - 2; i > 0; i--) {
                if(code[i][j] != 0) {
                    printf("%c", (char)code[i][j]);
                    j++;
                }
            }
        }
        count++;
    }
    printf("\n");
}
```

```
int main() {
    char str[1000];
    int rails, choice;

    printf("Enter a Secret Message\n");
    gets(str);

    printf("Enter number of rails\n");
    scanf("%d", &rails);

    printf("Choose an option:\n");
    printf("1. Encrypt Message\n");
    printf("2. Decrypt Message\n");
    scanf("%d", &choice);

    switch(choice) {
        case 1:
            encryptMessage(str, rails);
            break;
        case 2:
            decryptMessage(str, rails);
            break;
        default:
            printf("Invalid choice\n");
            break;
    }

    return 0;
}
```

Program Statement 4: Write a C program to implement encryption and decryption using Hill Cipher method.

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

int main()
{
   unsigned int a[3][3] = {{6,24,1}, {13,16,10}, {20,17,15}};
   unsigned int b[3][3] = {{8,5,10}, {21,8,21}, {21,12,8}};
   int i,j;

   unsigned int c[20],d[20];
   char msg[20];
   int determinant = 0,t=0;
   ;
   printf("Enter the plaintext: \n");
   scanf("%s", msg);
   for(i=0;i<3;i++){
      c[i] = msg[i] - 65;
      printf("%d", c[i]);
   }
   for(i=0;i<3;i++){
      t=0;
      for(j=0;j<3;j++){
         t= t + (a[i][j]*c[j]);
      }
      d[i] = t%26;
   }

   printf("\nencrypted cipher text: ");
   for(i=0;i<3;i++)
      printf("%c",d[i] + 65);
   for(i=0;i<3;i++){
      t=0;
      for(j=0;j<3;j++){
         t = t + (b[i][j]*d[j]);
      }
      c[i] = t%26;
   }
   printf("\ndecrypted cipher text: ");
   for(i=0;i<3;i++)
   {
      printf("%c", c[i]+65);
   }
   return 0;
}
```

OUTPUT:

```
Enter the plaintext:
SAN
18013
encrypted cipher text: RAJ
decrypted cipher text: SAN
```

PROGRAM - 05

```
#include<stdio.h>

int IP[] = {2, 6, 3, 1, 4, 8, 5, 7};

int IP_inverse[] = {4, 1, 3, 5, 7, 2, 8, 6};

int S0[4][4] = {
   {1, 0, 3, 2},
   {3, 2, 1, 0},
   {0, 2, 1, 3},
   {3, 1, 3, 2}
};

int S1[4][4] = {
   {0, 1, 2, 3},
   {2, 0, 1, 3},
   {3, 0, 1, 0},
   {2, 1, 0, 3}
};
int initial_permutation(int plaintext) {
   int result = 0;
   for (int i = 0; i < 8; i++) {
      result |= ((plaintext >> (8 - IP[i])) & 1) << (7 - i);
   }
   return result;
}

int inverse_initial_permutation(int ciphertext) {
   int result = 0;
   for (int i = 0; i < 8; i++) {
      result |= ((ciphertext >> (8 - IP_inverse[i])) & 1) << (7 - i);
   }
   return result;
}

int s_box_substitution(int value, int s_box[4][4]) {
   int row = ((value & 0b1000) >> 2) | (value & 0b0001);
   int col = (value & 0b0110) >> 1;
   return s_box[row][col];
}

int main() {
   int plaintext = 0b11010110;
   printf("Plain Text: %x\n", plaintext);

   int cipher_text = initial_permutation(plaintext);
   printf("Cipher Text: %x\n", cipher_text);

   // Example of S-box substitution
   int s_box_value = 0b1101; // Example value
   int s_box_result = s_box_substitution(s_box_value, S0);
   printf("S-box result: %x\n", s_box_result);

   int decrypted_text = inverse_initial_permutation(cipher_text);
   printf("Decrypted Text: %x\n", decrypted_text);

   return 0;
}
```