EXPERIMENT-1

Implement the data link layer framing methods such as character-stuffing and bit stuffing.

```
#include <stdio.h>
#include <string.h>
#define FLAG '$'    // Frame delimiter for Character Stuffing
#define ESC '\\'   // Escape character for Character Stuffing
// Function for Character Stuffing
void characterStuffing(char *input) {
    printf("Character Stuffed Output: %c", FLAG);
    for (int i = 0; i < strlen(input); i++) {
        if (input[i] == FLAG || input[i] == ESC) {
            printf("%c", ESC);
        }
        printf("%c", input[i]);
    }
    printf("%c\n", FLAG);
}
// Function for Bit Stuffing
void bitStuffing(char *input) {
    printf("Bit Stuffed Output: 01111110 "); // Flag sequence for bit stuffing
    int count = 0;
    for (int i = 0; i < strlen(input); i++) {
        if (input[i] == '1') {
```

```
            count++;
            printf("1");
            if (count == 5) { // After five consecutive 1s, insert a 0
                printf("0");
                count = 0;
            }
        } else {
            printf("0");
            count = 0;
        }
    }
    printf(" 01111110\n"); // Ending flag sequence
}

int main() {
    int choice;
    char input[100];

    printf("Choose Framing Method:\n1. Character Stuffing\n2. Bit Stuffing\nEnter your choice: ");
    scanf("%d", &choice);
    getchar();  // Consume newline character

    printf("Enter the data string: ");
```

```
    fgets(input, sizeof(input), stdin);
    input[strcspn(input, "\n")] = 0; // Remove newline character

    if (choice == 1) {
        characterStuffing(input);
    } else if (choice == 2) {
        bitStuffing(input);
    } else {
        printf("Invalid choice!\n");
    }

    return 0;
}
```

EXPERIMENT-2

Write a program to compute CRC code for the polynomials

```
// Include headers
#include<stdio.h>
#include<string.h>
// length of the generator polynomial
#define N strlen(gen_poly)
// data to be transmitted and received
char data[28];
// CRC value
char check_value[28];
// generator polynomial
char gen_poly[10];
// variables
int data_length,i,j;
// function that performs XOR operation
void XOR(){
   // if both bits are the same, the output is 0
   // if the bits are different the output is 1
   for(j = 1;j < N; j++)
   check_value[j] = (( check_value[j] == gen_poly[j])?'0':'1');
}
void crc(){
   // initializing check_value
```

```
    for(i=0;i<N;i++)
        check_value[i]=data[i];
    do{
    // check if the first bit is 1 and calls XOR function
        if(check_value[0]=='1')
            XOR();
// Move the bits by 1 position for the next computation
        for(j=0;j<N-1;j++)
            check_value[j]=check_value[j+1];
        // appending a bit from data
        check_value[j]=data[i++];
    }while(i<=data_length+N-1);
// loop until the data ends
}
// Function to check for errors on the receiver side
void receiver(){
// get the received data
    printf("Enter the received data: ");
   scanf("%s", data);
    printf("\n-----------------------------\n");
    printf("Data received: %s", data);
// Cyclic Redundancy Check
    crc();
// Check if the remainder is zero to find the error
```

```
    for(i=0;(i<N-1) && (check_value[i]!='1');i++);
if(i<N-1)
            printf("\nError detected\n\n");
        else
            printf("\nNo error detected\n\n");
}
int main()
{
    // get the data to be transmitted
    printf("\nEnter data to be transmitted: ");
    scanf("%s",data);
    printf("\n Enter the Generating polynomial: ");
    // get the generator polynomial
    scanf("%s",gen_poly);
    // find the length of data
    data_length=strlen(data);
    // appending n-1 zeros to the data
    for(i=data_length;i<data_length+N-1;i++)
        data[i]='0';
    printf("\n----------------------------------------");
// print the data with padded zeros
    printf("\n Data padded with n-1 zeros : %s",data);
    printf("\n----------------------------------------");
// Cyclic Redundancy Check
```

```
    crc();
// print the computed check value
    printf("\nCRC or Check value is : %s",check_value);
// Append data with check_value(CRC)
    for(i=data_length;i<data_length+N-1;i++)
        data[i]=check_value[i-data_length];
    printf("\n----------------------------------------");
// printing the final data to be sent
    printf("\n Final data to be sent : %s",data);
    printf("\n----------------------------------------\n");
// Calling the receiver function to check errors
    receiver();
        return 0;
}
```

EXPERIMENT-3

Develop a simple data link layer that performs the flow control using the sliding window protocol,and loss recovery using the Go-Back-N mechanism.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <time.h>
#define WINDOW_SIZE 4 // Size of the sliding window
#define TOTAL_FRAMES 10 // Total number of frames to be sent
#define TIMEOUT 3 // Timeout duration in seconds
int send_frame(int frame) {
    // Simulate random loss of frames (e.g., frame 2 is lost)
    if (rand() % 5 == 0) {
        printf("[Sender] Frame %d lost during transmission.\n", frame);
        return 0; // Simulate frame loss
    }
    printf("[Sender] Frame %d sent successfully.\n", frame);
    return 1; // Frame sent successfully
}
int receive_ack(int frame) {
    // Simulate random loss of ACKs (e.g., ACK for frame 3 is lost)
    if (rand() % 7 == 0) {
        printf("[Receiver] ACK for Frame %d lost.\n", frame);
```

```
        return 0; // Simulate ACK loss
    }
    printf("[Receiver] ACK for Frame %d received successfully.\n", frame);
    return 1; // ACK received successfully
}
void sliding_window_protocol() {
    int base = 0;      // Oldest unacknowledged frame
    int next_frame = 0; // Next frame to be sent
    int acks[TOTAL_FRAMES] = {0}; // Array to track acknowledgments
    time_t timers[TOTAL_FRAMES]; // Timers for each frame
    while (base < TOTAL_FRAMES) {
        // Send frames within the window
        while (next_frame < base + WINDOW_SIZE && next_frame < TOTAL_FRAMES)
{
            if (!acks[next_frame]) {
                send_frame(next_frame);
                timers[next_frame] = time(NULL); // Start the timer
            }
            next_frame++;
        }
        // Check for ACKs or timeouts
        for (int i = base; i < next_frame; i++) {
            if (acks[i]) {
                continue; // Skip acknowledged frames
```

```
            }
            // Simulate receiving an acknowledgment
            if (receive_ack(i)) {
                acks[i] = 1; // Mark frame as acknowledged
                // Slide the window if the base frame is acknowledged
                while (base < TOTAL_FRAMES && acks[base]) {
                    printf("[Sender] Sliding window. Base frame is now %d.\n", base + 1)
            base++;
                }
            } else {
                // Check for timeout
                if (difftime(time(NULL), timers[i]) > TIMEOUT) {
                    printf("[Sender] Timeout for Frame %d. Resending all frames from %d.\n", i,
base);
                    // Resend all frames from the base frame
                    next_frame = base;
                    break;
                }
            }
        }
    }
    printf("[Sender] All frames sent and acknowledged successfully.\n");
}
int main() {
```

```
    srand(time(NULL)); // Seed for random number generation

    printf("[Sender] Starting Sliding Window Protocol with Go-Back-N.\n");
    sliding_window_protocol();
    printf("[Sender] Transmission completed.\n");
    return 0;
}
```

EXPERIMENT-4

Take an example subnet of hosts and obtain a broadcast tree for the subnet.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
// Function to convert an IP address to a 32-bit integer
unsigned int ipToInt(char* ip) {
    unsigned int a, b, c, d;
    sscanf(ip, "%u.%u.%u.%u", &a, &b, &c, &d);
    return (a << 24) | (b << 16) | (c << 8) | d;
}
// Function to convert a 32-bit integer to an IP address
void intToIp(unsigned int ip, char* buffer) {
    sprintf(buffer, "%u.%u.%u.%u", (ip >> 24) & 0xFF, (ip >> 16) & 0xFF, (ip
>> 8) & 0xFF, ip & 0xFF);
}
// Function to calculate the subnet mask from a prefix length
unsigned int calculateSubnetMask(int prefixLength) {
    return prefixLength == 0 ? 0 : ~((1 << (32 - prefixLength)) - 1);
}
int main() {
    char ip[16];
    int prefixLength, newPrefixLength;
    unsigned int subnetMask, newSubnetMask, ipInt;
    char buffer[16];
```

```
    // Input IP address and prefix length
    printf("Enter IP address (e.g., 192.168.1.0): ");
    scanf("%s", ip);
    printf("Enter current prefix length (e.g., 24): ");
    scanf("%d", &prefixLength);
    // New prefix length for creating two subnets
    newPrefixLength = prefixLength + 1;
    // Convert IP address to integer
    ipInt = ipToInt(ip);
    // Calculate original subnet mask and new subnet mask
    subnetMask = calculateSubnetMask(prefixLength);
    newSubnetMask = calculateSubnetMask(newPrefixLength);
    // Calculate the number of hosts per subnet
    int hostsPerSubnet = (1 << (32 - newPrefixLength)) - 2; // subtract 2 for
network and broadcast addresses
    printf("\nNumber of subnets: 2\n");
    printf("Number of hosts per subnet: %d\n", hostsPerSubnet);
    // Generate subnets
    for (int i = 0; i < 2; i++) {
        unsigned int subnetNetwork = (ipInt & subnetMask) | (i << (32 -
newPrefixLength));
        unsigned int subnetBroadcast = subnetNetwork | ~newSubnetMask;
        unsigned int firstHost = subnetNetwork + 1;
        unsigned int lastHost = subnetBroadcast - 1;
        printf("\nSubnet %d:\n", i + 1);
```

```
        printf("Network Address: ");
        intToIp(subnetNetwork, buffer);
        printf("%s\n", buffer);
        printf("Broadcast Address: ");
        intToIp(subnetBroadcast, buffer);
        printf("%s\n", buffer);
        printf("Subnet Mask: ");
        intToIp(newSubnetMask, buffer);
        printf("%s\n", buffer);
        printf("First Host: ");
        intToIp(firstHost, buffer);
        printf("%s\n", buffer);
        printf("Last Host: ");
        intToIp(lastHost, buffer);
        printf("%s\n", buffer);
    }
    return 0;
}
```

EXPERIMENT-5

Implement distance vector routing algorithm for obtaining routing tables at each node.

```
#include <stdio.h>

#include <stdlib.h>

#define INF 9999

#define MAX_NODES 10

// Function to initialize distance vector and routing table

void initialize(int numNodes, int costMatrix[MAX_NODES][MAX_NODES],
int distVector[MAX_NODES][MAX_NODES], int
nextHop[MAX_NODES][MAX_NODES]) {

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

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

distVector[i][j] = costMatrix[i][j];

if (costMatrix[i][j] != INF && i != j) {

nextHop[i][j] = j;

} else {

nextHop[i][j] = -1;

}

}

}

}

// Function to print routing table for each node

void printRoutingTable(int numNodes, int
distVector[MAX_NODES][MAX_NODES], int
nextHop[MAX_NODES][MAX_NODES]) {
```

```
for (int i = 0; i < numNodes; i++) {
printf("Routing table for node %d:\n", i);
printf("Destination\tNext Hop\tDistance\n");
for (int j = 0; j < numNodes; j++) {
if (distVector[i][j] == INF) {
printf("%d\t\t-\t\tINF\n", j);
} else {
printf("%d\t\t%d\t\t%d\n", j, nextHop[i][j], distVector[i][j]);
}
}
printf("\n");
}
}
// Function to implement Distance Vector Routing algorithm
void distanceVectorRouting(int numNodes, int
costMatrix[MAX_NODES][MAX_NODES], int
distVector[MAX_NODES][MAX_NODES], int
nextHop[MAX_NODES][MAX_NODES]) {
int updated;
do {
updated = 0;
for (int i = 0; i < numNodes; i++) {
for (int j = 0; j < numNodes; j++) {
for (int k = 0; k < numNodes; k++) {
if (distVector[i][k] + distVector[k][j] < distVector[i][j]) {
```

```
distVector[i][j] = distVector[i][k] + distVector[k][j];
nextHop[i][j] = nextHop[i][k];
updated = 1;
}
}
}       }
            } while (updated);
        }
        int main() {
            int numNodes, costMatrix[MAX_NODES][MAX_NODES];
            int distVector[MAX_NODES][MAX_NODES];
            int nextHop[MAX_NODES][MAX_NODES];
            printf("Enter the number of nodes: ");
            scanf("%d", &numNodes);
            printf("Enter the cost matrix (use %d for INF):\n", INF);
            for (int i = 0; i < numNodes; i++) {
                for (int j = 0; j < numNodes; j++) {
                    scanf("%d", &costMatrix[i][j]);
                }
            }
            initialize(numNodes, costMatrix, distVector, nextHop);
            distanceVectorRouting(numNodes, costMatrix, distVector, nextHop);
            printRoutingTable(numNodes, distVector, nextHop);
            return 0;     }
```

EXPERIMENT-6

Implement Dijkstra's algorithm to compute the shortest path through a network.

```
#include <stdio.h>

#define INFINITY 9999

#define MAX 10


void Dijkstra(int Graph[MAX][MAX], int n, int start) {
   int cost[MAX][MAX], distance[MAX], pred[MAX];
   int visited[MAX], count, mindistance, nextnode, i, j;


   // Create cost matrix
   for (i = 0; i < n; i++)
      for (j = 0; j < n; j++)
         cost[i][j] = (Graph[i][j] == 0) ? INFINITY : Graph[i][j];


   for (i = 0; i < n; i++) {
      distance[i] = cost[start][i];
      pred[i] = start;
      visited[i] = 0;
   }


   distance[start] = 0;
   visited[start] = 1;
```

```
    count = 1;

    while (count < n - 1) {
        mindistance = INFINITY;
        for (i = 0; i < n; i++)
            if (distance[i] < mindistance && !visited[i]) {
                mindistance = distance[i];
                nextnode = i;
            }
        visited[nextnode] = 1;
        for (i = 0; i < n; i++)
            if (!visited[i])
                if (mindistance + cost[nextnode][i] < distance[i]) {
                    distance[i] = mindistance + cost[nextnode][i];
                    pred[i] = nextnode;
                }
        count++;
    }

    for (i = 0; i < n; i++)
        if (i != start)
            printf("\nDistance from source to %d: %d", i, distance[i]);
}
```

```
int main() {
    int Graph[MAX][MAX] = {
        {0, 0, 1, 2, 0, 0, 0},
        {0, 0, 2, 0, 0, 3, 0},
        {1, 2, 0, 1, 3, 0, 0},
        {2, 0, 1, 0, 0, 0, 1},
        {0, 0, 3, 0, 0, 2, 0},
        {0, 3, 0, 0, 2, 0, 1},
        {0, 0, 0, 1, 0, 1, 0}
    };
    int n = 7, u = 0;
    Dijkstra(Graph, n, u);
    return 0;
}
```

EXPERIMENT-7

Write a program for congestion control using Leaky bucket algorithm.

```
#include <stdio.h>

int main() {
   int bucket_size, output_rate, input_packets, stored = 0, time;

   printf("Enter bucket size: ");
   scanf("%d", &bucket_size);

   printf("Enter output rate (leak rate): ");
   scanf("%d", &output_rate);

   printf("Enter number of cycles (time units): ");
   scanf("%d", &time);

   while (time--) {
      printf("\nEnter number of packets arriving: ");
      scanf("%d", &input_packets);

      // Check for overflow condition
      if (stored + input_packets > bucket_size) {
         printf("Bucket overflow! %d packets dropped.\n", (stored +
input_packets) - bucket_size);
         stored = bucket_size;
      } else {
         stored += input_packets;
      }

      printf("Packets currently in bucket: %d\n", stored);

      // Leak / Send packets
      if (stored >= output_rate) {
         stored -= output_rate;
         printf("Packets sent: %d\n", output_rate);
      } else {
```

```
            printf("Packets sent: %d\n", stored);
            stored = 0;
        }

        printf("Packets left in bucket after sending: %d\n", stored);
    }

    return 0;
}
```

EXPERIMENT-8

Implementation of DNS.

```
import java.net.*;
import java.util.Scanner;
public class SimpleDNSResolver {
 public static void main(String[] args) {
 Scanner scanner = new Scanner(System.in);
 System.out.println("Welcome to the Simple DNS Resolver!");
 System.out.println("Enter a domain name to resolve (or type 'exit' to quit):");
 while (true) {
 System.out.print("Domain: ");
 String domain = scanner.nextLine();
 if ("exit".equalsIgnoreCase(domain)) {
 System.out.println("Exiting DNS Resolver. Goodbye!");
 break; }
 try {
 InetAddress[] addresses = InetAddress.getAllByName(domain);
 System.out.println("IP addresses for " + domain + ":");
 for (InetAddress address : addresses) {
 System.out.println(" - " + address.getHostAddress());
 }
 } catch (UnknownHostException e) {
 System.out.println("Could not resolve domain: " + e.getMessage());}  }
 scanner.close();}}
```

EXPERIMENT-9

Implementation of Ping service.

```
import java.net.*;

import java.io.*;

public class PingService {

   public static void main(String[] args) {

      if (args.length != 1) {

         System.out.println("Usage: java PingService <hostname>");

         return;  }

      String hostname = args[0]; // Get the hostname from command-line
arguments

      try {

         System.out.println("Pinging " + hostname + "...");

         InetAddress inetAddress = InetAddress.getByName(hostname);

         boolean isReachable = inetAddress.isReachable(5000);

         if (isReachable) {

            System.out.println("Host " + hostname + " is reachable.");

            System.out.println("IP Address: " + inetAddress.getHostAddress());

         } else {

            System.out.println("Host " + hostname + " is not reachable.");   }    }
catch (UnknownHostException e) {

         System.out.println("Unknown host: " + hostname);

      } catch (IOException e) {

System.out.println("Error occurred while pinging " + hostname + ": " +
e.getMessage());      }    }   }
```

EXPERIMENT-10

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.

```
#include <stdio.h>

int main() {
    char str[100];   // Buffer to store user input
    int i = 0;

    printf("Enter a string: ");
    fgets(str, sizeof(str), stdin);  // Read input including spaces

    // XOR each character with 0 and display the result
    while (str[i] != '\0') {
        printf("%c", str[i] ^ 0);   // XOR with 0 (no change)
        i++;
    }

    return 0;
}
```

EXPERIMENT-11

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[100];
    int i;
    // Taking user input
    printf("Enter a string: ");
    fgets(str, sizeof(str), stdin);  // reads a line of text including spaces
    printf("\nOriginal String: %s\n", str);
    printf("Character\tASCII\tAND(127)\tXOR(127)\n");
    printf("-----------------------------------------\n");
    // Processing each character
    for (i = 0; str[i] != '\0'; i++) {
        if (str[i] == '\n')  // skip newline character from fgets
            continue;
        char and_result = str[i] & 127;  // Bitwise AND
        char xor_result = str[i] ^ 127;  // Bitwise XOR
        printf("    %c\t\t%d\t   %d\t\t   %d\n", str[i], str[i], and_result, xor_result);
    }
    return 0;
}
```

EXPERIMENT-12

Write a Java program to perform encryption and decryption using the following algorithms

a. **Ceaser cipher**

```
import java.util.Scanner;

public class CaesarCipher {

    // Method to encrypt text

    public static String encrypt(String text, int key) {

        StringBuilder result = new StringBuilder();

        for (char ch : text.toCharArray()) {

            if (Character.isUpperCase(ch)) {

                char c = (char) ((ch - 'A' + key) % 26 + 'A');

                result.append(c);

            } else if (Character.isLowerCase(ch)) {

                char c = (char) ((ch - 'a' + key) % 26 + 'a');

                result.append(c);

            } else {

                result.append(ch); // keep symbols/spaces unchanged

            }

        }

        return result.toString();

    }

    // Method to decrypt text

    public static String decrypt(String cipher, int key) {

        return encrypt(cipher, 26 - (key % 26)); // reverse shift
```

```
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
       System.out.print("Enter text: ");
        String text = sc.nextLine();
        System.out.print("Enter key (shift value): ");
        int key = sc.nextInt();
        String encrypted = encrypt(text, key);
        String decrypted = decrypt(encrypted, key);
        System.out.println("\n--- Results ---");
        System.out.println("Encrypted Text: " + encrypted);
        System.out.println("Decrypted Text: " + decrypted);
    }
}
```

b. **Substitution cipher**

```
import java.util.*;
public class SubstitutionCipher {
    // Method to encrypt text using substitution key
    public static String encrypt(String text, String key) {
        StringBuilder result = new StringBuilder();
        key = key.toUpperCase();
        for (char ch : text.toCharArray()) {
            if (Character.isUpperCase(ch)) {
                int index = ch - 'A';
                result.append(key.charAt(index));
            } else if (Character.isLowerCase(ch)) {
                int index = ch - 'a';
                result.append(Character.toLowerCase(key.charAt(index)));
            } else {
                result.append(ch); // keep spaces, numbers, symbols
            }
        }
        return result.toString();
    }
    // Method to decrypt text using substitution key
    public static String decrypt(String cipher, String key) {
        StringBuilder result = new StringBuilder();
        key = key.toUpperCase();
```

```
        // Create reverse mapping
        char[] reverseKey = new char[26];
        for (int i = 0; i < 26; i++) {
            reverseKey[key.charAt(i) - 'A'] = (char) ('A' + i);
        }
        for (char ch : cipher.toCharArray()) {
            if (Character.isUpperCase(ch)) {
                int index = ch - 'A';
                result.append(reverseKey[index]);
            } else if (Character.isLowerCase(ch)) {
                int index = Character.toUpperCase(ch) - 'A';
                result.append(Character.toLowerCase(reverseKey[index]));
            } else {
                result.append(ch);
            }
        }
        return result.toString();
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter 26-letter substitution key (e.g.,
QWERTYUIOPASDFGHJKLZXCVBNM): ");
        String key = sc.nextLine();
        // Validate key
        if (key.length() != 26 || !key.matches("[A-Za-z]+")) {
```

```
            System.out.println("Invalid key! Key must be 26 alphabetic characters.");
            return;
        }
        System.out.print("Enter text to encrypt: ");
        String text = sc.nextLine();
        String encrypted = encrypt(text, key);
  String decrypted = decrypt(encrypted, key);
        System.out.println("\n--- Results ---");
        System.out.println("Key Used:         " + key);
        System.out.println("Encrypted Text:   " + encrypted);
        System.out.println("Decrypted Text:   " + decrypted);
    }
}
```

c. **Hill Cipher**

```
import java.util.Scanner;
public class HillCipher {
    // Function to multiply matrices (2x2 key with 2x1 vector)
    private static int[] multiplyMatrix(int[][] key, int[] vector) {
        int[] result = new int[2];
        result[0] = (key[0][0] * vector[0] + key[0][1] * vector[1]) % 26;
        result[1] = (key[1][0] * vector[0] + key[1][1] * vector[1]) % 26;
        return result;
    }
    // Function to find modular inverse of determinant mod 26
    private static int modInverse(int det) {
        det = det % 26;
        for (int x = 1; x < 26; x++) {
            if ((det * x) % 26 == 1)
                return x;
        }
        return -1; // inverse doesn't exist
    }
    // Function to compute inverse of 2x2 key matrix
    private static int[][] invertKey(int[][] key) {
        int det = key[0][0] * key[1][1] - key[0][1] * key[1][0];
        det = (det % 26 + 26) % 26;
        int invDet = modInverse(det);
```

```
        if (invDet == -1) {
            throw new RuntimeException("Inverse doesn't exist for this key matrix!");
        }
        int[][] invKey = new int[2][2];
        invKey[0][0] = (key[1][1] * invDet) % 26;
        invKey[1][1] = (key[0][0] * invDet) % 26;
        invKey[0][1] = ((-key[0][1] + 26) * invDet) % 26;
        invKey[1][0] = ((-key[1][0] + 26) * invDet) % 26;
        return invKey;
    }
    // Encrypt function
    public static String encrypt(String plaintext, int[][] key) {
        plaintext = plaintext.toUpperCase().replaceAll("[^A-Z]", "");
        if (plaintext.length() % 2 != 0)
            plaintext += "X";
        StringBuilder cipher = new StringBuilder();
        for (int i = 0; i < plaintext.length(); i += 2) {
            int[] vector = {
                plaintext.charAt(i) - 'A',
                plaintext.charAt(i + 1) - 'A'
            };
            int[] result = multiplyMatrix(key, vector);
            cipher.append((char) (result[0] + 'A'));
            cipher.append((char) (result[1] + 'A'));
```

```
        }
        return cipher.toString();
    }
    // Decrypt function
    public static String decrypt(String ciphertext, int[][] key) {
        ciphertext = ciphertext.toUpperCase().replaceAll("[^A-Z]", "");
int[][] invKey = invertKey(key);
StringBuilder plain = new StringBuilder();
        for (int i = 0; i < ciphertext.length(); i += 2) {
            int[] vector = {
                ciphertext.charAt(i) - 'A',
                ciphertext.charAt(i + 1) - 'A'
            };
    int[] result = multiplyMatrix(invKey, vector);
            plain.append((char) (result[0] + 'A'));
            plain.append((char) (result[1] + 'A'));
        }
        return plain.toString();
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Hill Cipher (2x2 Matrix)");
        System.out.println("Enter 2x2 key matrix (values 0–25):");
        int[][] key = new int[2][2];
```

```
        for (int i = 0; i < 2; i++)
            for (int j = 0; j < 2; j++)
                key[i][j] = sc.nextInt();
        sc.nextLine(); // clear buffer
        System.out.print("Enter plaintext: ");
        String plaintext = sc.nextLine();
        String encrypted = encrypt(plaintext, key);
        String decrypted = decrypt(encrypted, key);
        System.out.println("\n--- Results ---");
        System.out.println("Encrypted Text: " + encrypted);
        System.out.println("Decrypted Text: " + decrypted);
        sc.close();
    }
}
```