Government College of Engineering, Jalgaon
(An Autonomous Institute of Govt. of Maharashtra)

Name: **PRN:**

Class: L.Y **Semester:** VII **Batch:** B

Date of Performance: _____ **Date of Completion:** _____

Subject: CO407U CNSL **Subject Teacher:** Ms. Shrutika Mahajan

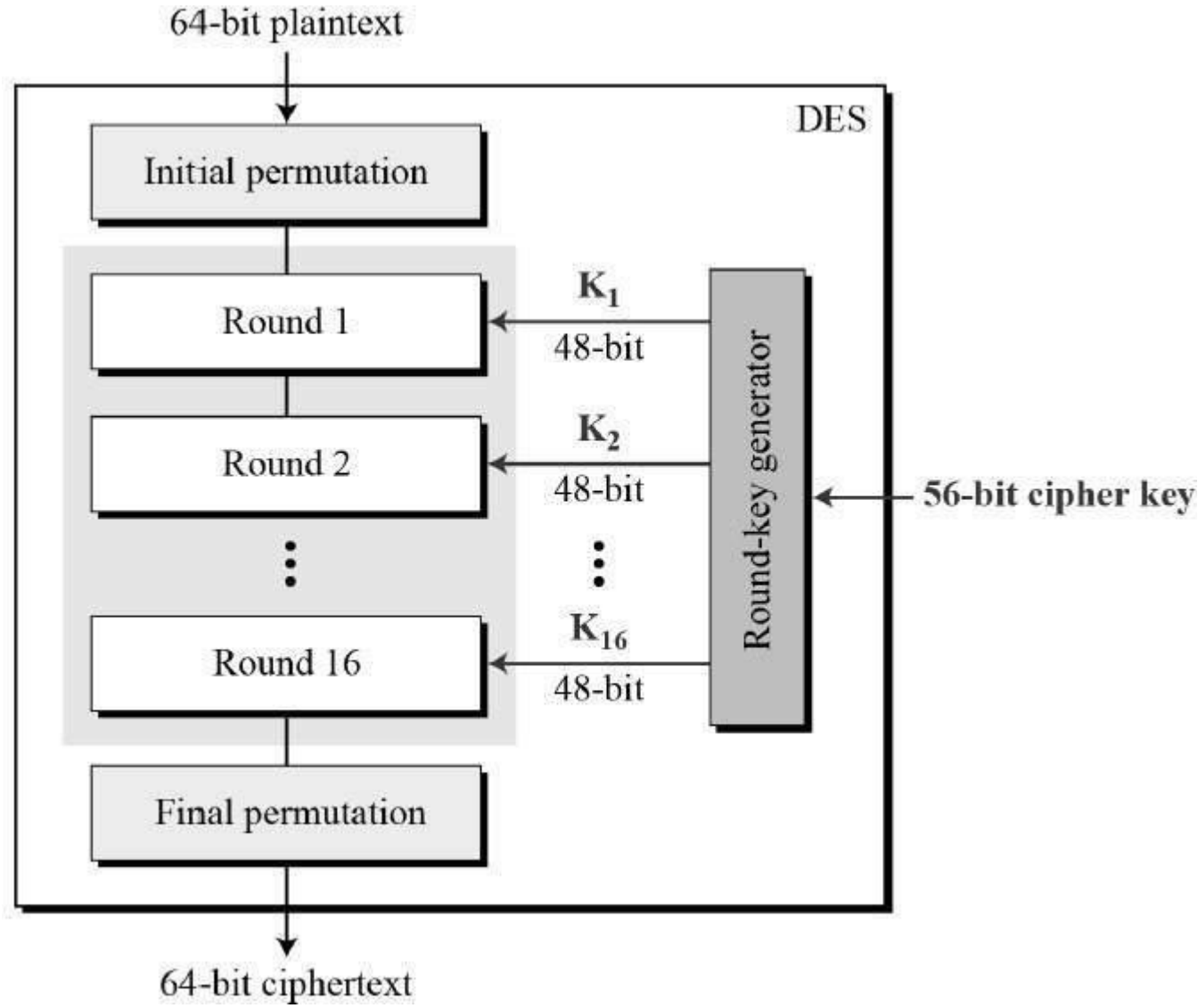
Practical – 7

Aim: Implement DES Algorithm

Theory:

The Data Encryption Standard (DES) is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST).

DES is an implementation of a Feistel Cipher. It uses 16 round Feistel structure. The block size is 64-bit. Though, key length is 64-bit, DES has an effective key length of 56 bits, since 8 of the 64 bits of the key are not used by the encryption algorithm (function as check bits only). General Structure of DES is depicted in the following illustration −

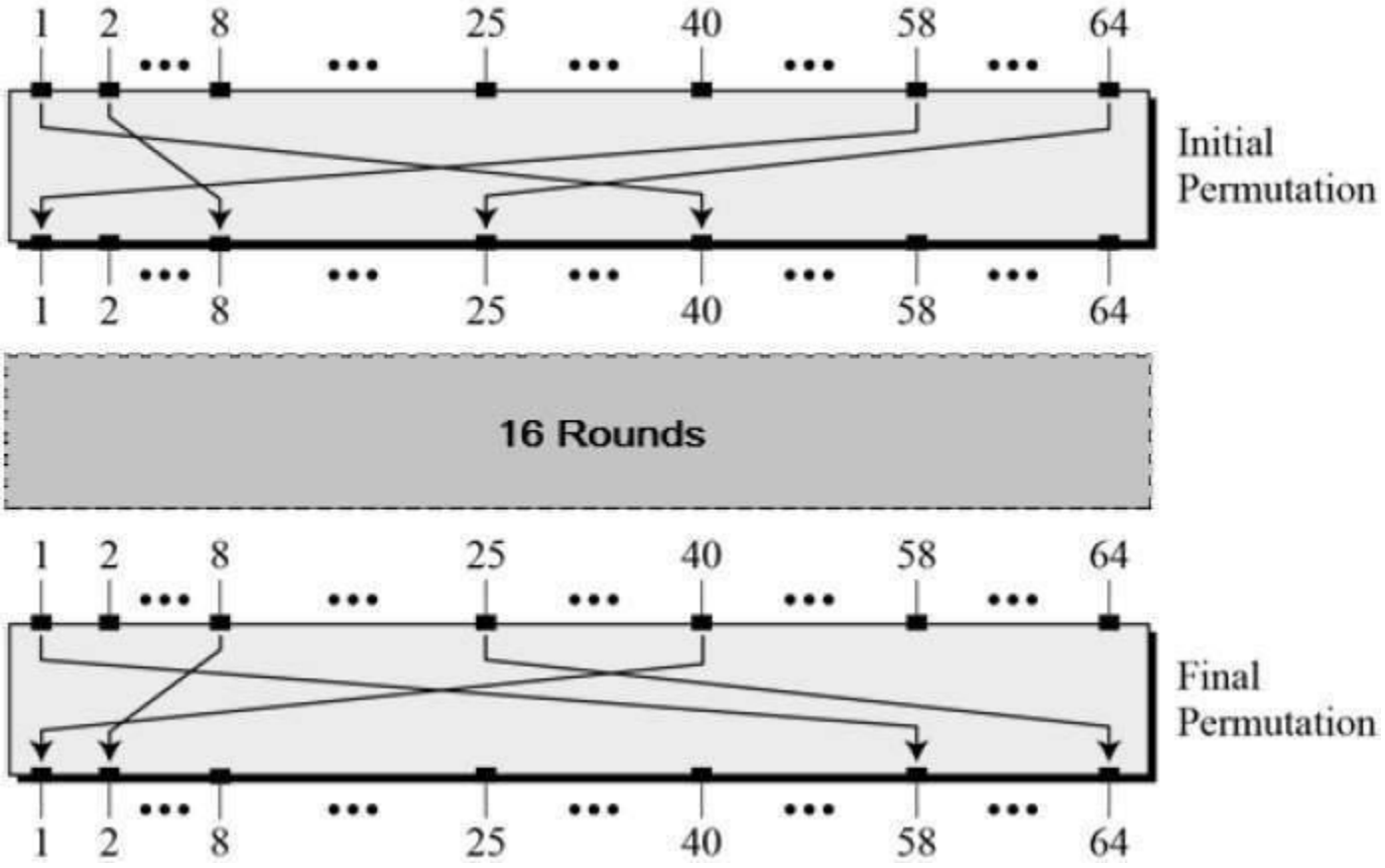


Since DES is based on the Feistel Cipher, all that is required to specify DES is −

- Round function
- Key schedule
- Any additional processing − Initial and final permutation

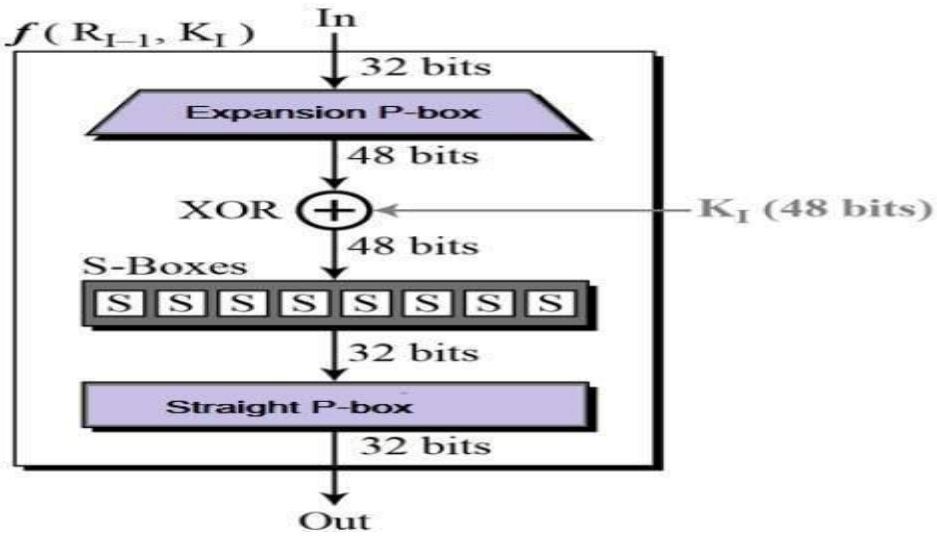
Initial and Final Permutation

The initial and final permutations are straight Permutation boxes (P-boxes) that are inverses of each other. They have no cryptography significance in DES. The initial and final permutations are shown as follows −

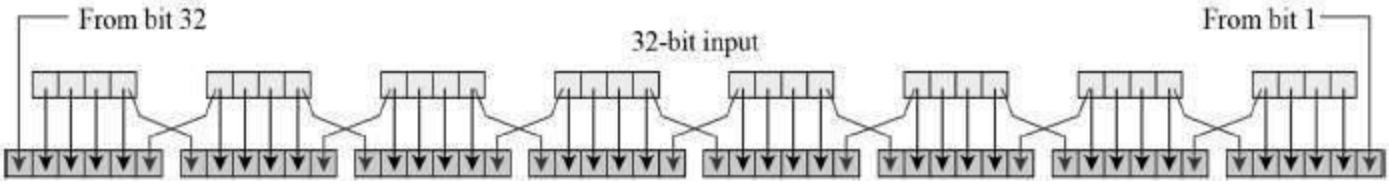


Round Function

The heart of this cipher is the DES function, f. The DES function applies a 48-bit key to the rightmost 32 bits to produce a 32-bit output.



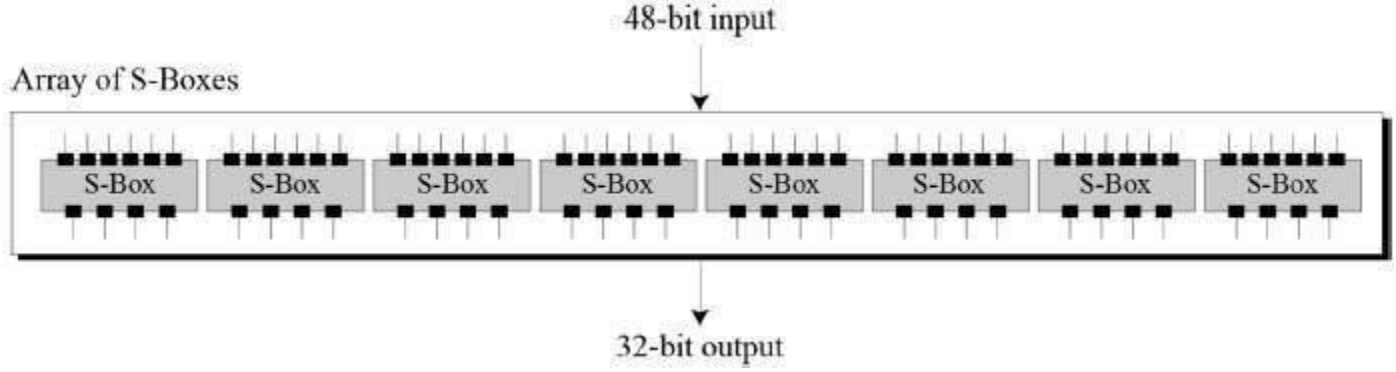
- **Expansion Permutation Box** − Since right input is 32-bit and round key is a 48-bit, we first need to expand right input to 48 bits. Permutation logic is graphically depicted in the following illustration –



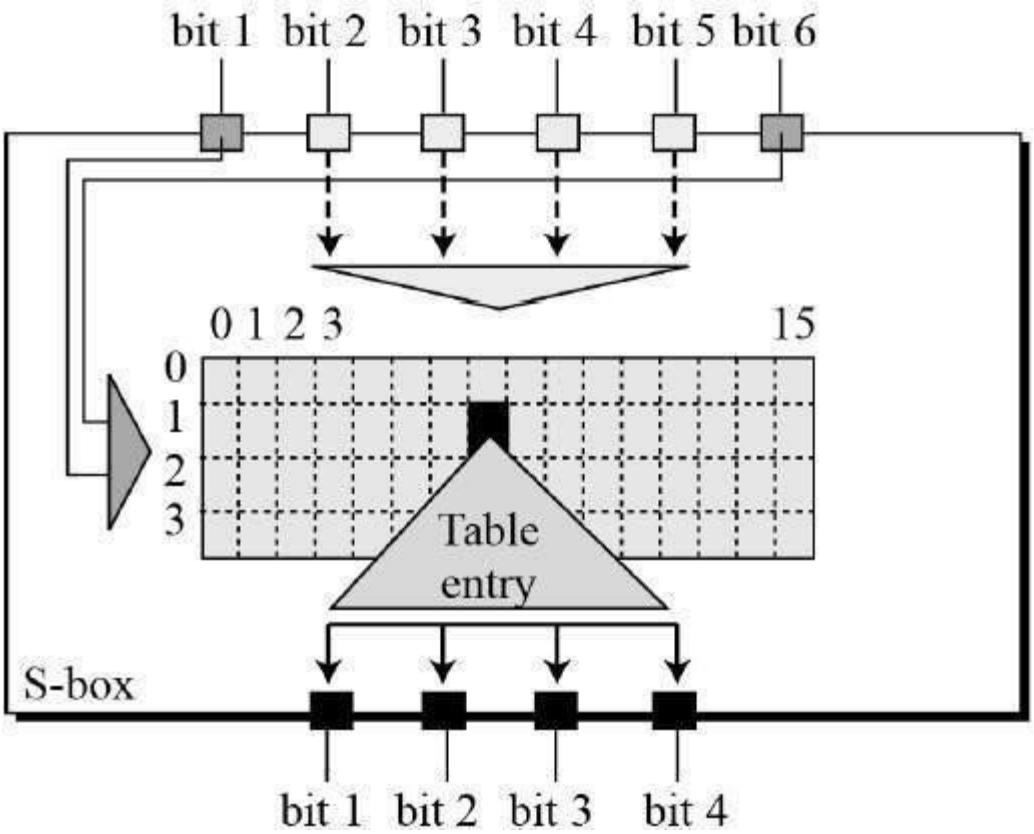
- The graphically depicted permutation logic is generally described as table in DES specification illustrated as shown –

32	01	02	03	04	05
04	05	06	07	08	09
08	09	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	31	31	32	01

- **XOR (Whitener).** − After the expansion permutation, DES does XOR operation on the expanded right section and the round key. The round key is used only in this operation.
- **Substitution Boxes.** − The S-boxes carry out the real mixing (confusion). DES uses 8 S-boxes, each with a 6-bit input and a 4-bit output. Refer the following illustration −



- The S-box rule is illustrated below −

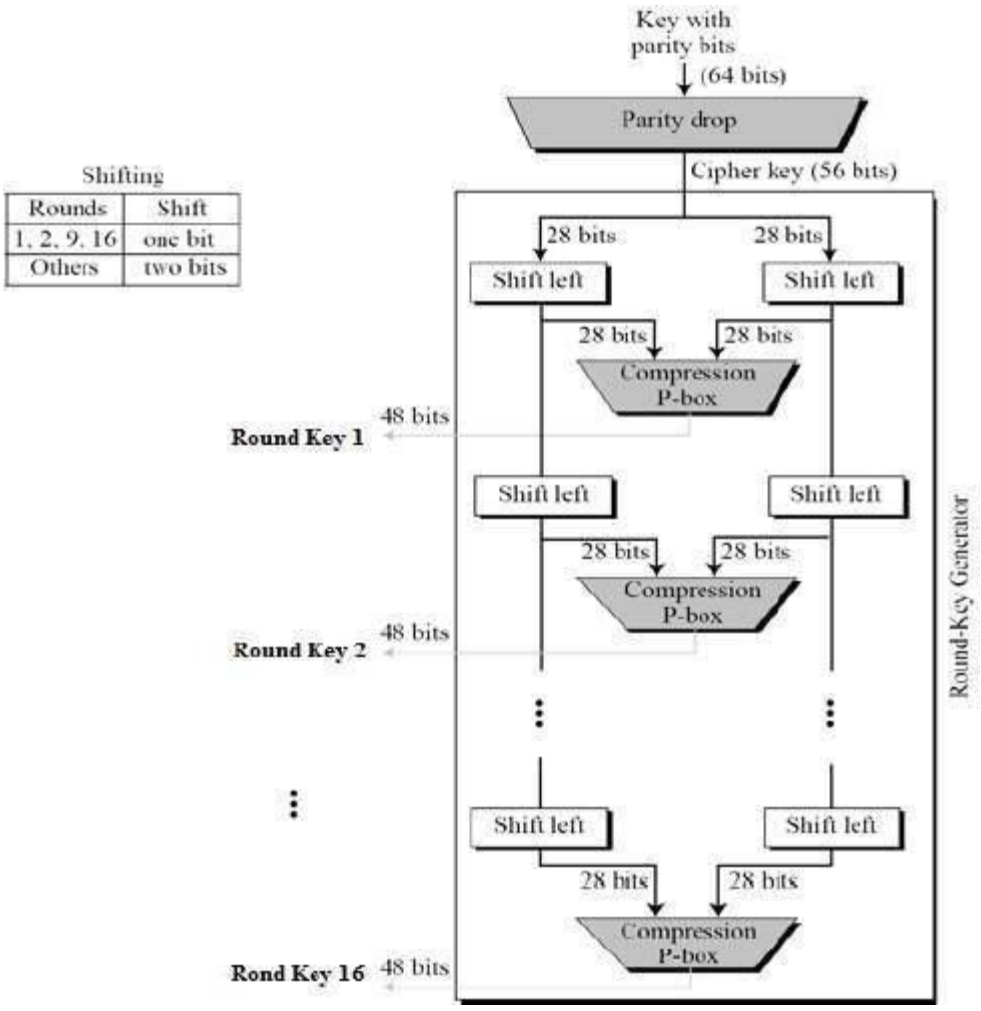


- There are a total of eight S-box tables. The output of all eight s-boxes is then combined in to 32 bit section.
- **Straight Permutation** − The 32 bit output of S-boxes is then subjected to the straight permutation with rule shown in the following illustration:

16	07	20	21	29	12	28	17
01	15	23	26	05	18	31	10
02	08	24	14	32	27	03	09
19	13	30	06	22	11	04	25

Key Generation

The round-key generator creates sixteen 48-bit keys out of a 56-bit cipher key. The process of key generation is depicted in the following illustration −



The logic for Parity drop, shifting, and Compression P-box is given in the DES description.

DES Analysis

The DES satisfies both the desired properties of block cipher. These two properties make cipher very strong.

- **Avalanche effect** − A small change in plaintext results in the very great change in the ciphertext.
- **Completeness** − Each bit of ciphertext depends on many bits of plaintext.

During the last few years, cryptanalysis have found some weaknesses in DES when key selected are weak keys. These keys shall be avoided.

DES has proved to be a very well designed block cipher. There have been no significant cryptanalytic attacks on DES other than exhaustive key search.

Code:

```
import java.util.Scanner;
public class des {
    static int charToNumber(char c) {
        if (Character.isUpperCase(c)) return c - 'A';
        if (Character.isLowerCase(c)) return c - 'a' + 26;
        if (Character.isDigit(c)) return c - '0' + 52;
        return c % 62;
    }
    static char numberToChar(int n) {
        n = ((n % 62) + 62) % 62;
        if (n < 26) return (char) ('A' + n);
        else if (n < 52) return (char) ('a' + (n - 26));
        else return (char) ('0' + (n - 52));
    }
    static int mangler(int value, int keyVal, int round) {
        return (value * 7 + keyVal * 5 + round * 3) % 62;
    }
    static final int INV_7_MOD_62 = 9;
    static String encryptRound(String text, String key, int round) {
        System.out.println("\n==============================");
        System.out.println("         ROUND " + round + " WORKING");
        System.out.println("==============================");
        System.out.println(String.format("%-10s %-10s %-25s %-10s", "Char", "Key",
"Mangler(value,key,round)", "ResultNum"));
        System.out.println("----------------------------------------");
        StringBuilder result = new StringBuilder();
        for (int i = 0; i < text.length(); i++) {
```

```
        char plainChar = text.charAt(i);
        char keyChar = key.charAt(i % key.length());
        int t = charToNumber(plainChar);
        int k = charToNumber(keyChar);
        int mixed = mangler(t, k, round);
        System.out.println(String.format("%-10s %-10s %-25s %-10s",
                plainChar, keyChar, "(" + t + "," + k + "," + round + ")", mixed));
        result.append(numberToChar(mixed));
    }
    String permuted = new StringBuilder(result.toString()).reverse().toString();
    System.out.println("After Round " + round + ": " + permuted);
    return permuted;
}
static String encrypt(String text, String key) {
    String result = text;
    for (int i = 1; i <= 3; i++) {
        result = encryptRound(result, key, i);
    }
    return result;
}
static String decrypt(String text, String key) {
    String result = text;
    for (int i = 3; i >= 1; i--) {
        result = new StringBuilder(result).reverse().toString();
        StringBuilder temp = new StringBuilder();
        for (int j = 0; j < result.length(); j++) {
            int c = charToNumber(result.charAt(j));
            int k = charToNumber(key.charAt(j % key.length()));
```

```
            // inverse: t = inv7 * ( c - (5*k + 3*r) ) mod 62
            int subtract = (5 * k + 3 * i) % 62;
            int diff = (c - subtract) % 62;
            if (diff < 0) diff += 62;
            int original = (INV_7_MOD_62 * diff) % 62;
            temp.append(numberToChar(original));
        }
        result = temp.toString();
    }
    return result;
  }
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.println("===== USER DEFINED DES ALGORITHM (3 ROUNDS) =====");
    System.out.print("Enter Plain Text (letters/digits only): ");
    String plaintext = sc.nextLine();
    System.out.print("Enter Key (letters/digits only): ");
    String key = sc.nextLine();
    String encrypted = encrypt(plaintext, key);
    String decrypted = decrypt(encrypted, key);

    System.out.println("\n===== FINAL RESULTS =====");
    System.out.println("Plaintext : " + plaintext);
    System.out.println("Key       : " + key);
    System.out.println("Encrypted : " + encrypted);
    System.out.println("Decrypted : " + decrypted);
  }
}
```

Output:

```
PS D:\Me\Computer\Study Material\CNS Practi\Program> javac des.java
PS D:\Me\Computer\Study Material\CNS Practi\Program> java des
===== USER DEFINED DES ALGORITHM (3 ROUNDS) =====
Enter Plain Text (letters/digits only): HELLO
Enter Key (letters/digits only): AVi

============================
        ROUND 1 WORKING
============================
Char       Key        Mangler(value,key,round)  ResultNum

H          A          (7,0,1)                   52
E          V          (4,21,1)                  12
L          i          (11,34,1)                 2
L          A          (11,0,1)                  18
O          V          (14,21,1)                 20
After Round 1: USCM0

============================
        ROUND 2 WORKING
============================
Char       Key        Mangler(value,key,round)  ResultNum

U          A          (20,0,2)                  22
S          V          (18,21,2)                 51
C          i          (2,34,2)                  4
M          A          (12,0,2)                  28
0          V          (52,21,2)                 41
After Round 2: pcEzW

============================
        ROUND 3 WORKING
============================
Char       Key        Mangler(value,key,round)  ResultNum

p          A          (41,0,3)                  48
c          V          (28,21,3)                 0
E          i          (4,34,3)                  21
z          A          (51,0,3)                  56
W          V          (22,21,3)                 20
After Round 3: U4VAw

===== FINAL RESULTS =====
Plaintext : HELLO
Key       : AVi
Encrypted : U4VAw
Decrypted : HELLO
```

Conclusion:

The Data Encryption Standard (DES) is one of the earliest and most influential symmetric-key encryption algorithms. It operates on 64-bit blocks of data using a 56-bit key and performs 16 rounds of substitution and permutation to provide data confidentiality.

Course Teacher
Ms. Shrutika Mahajan