#CNS (20CP320P)

Name: Jatin Prajapati

Roll No: 21BCP452D

Semester: 6th

Division: 5th (G-10)

PANDIT DEENDAYAL ENERGY UNIVERSITY

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

Practical 1

1. Implementation of SDES algorithm in C language.

Keygen.h

```c
// HEADER FILE FOR KEY GENERATION

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

void permuteKey(char key[], char permutedKey[])
{
    int i;
    int permutationTable[10] = {3, 5, 2, 7, 4, 10, 1, 9, 8, 6};

    for (i = 0; i < 10; i++)
    {
        permutedKey[i] = key[permutationTable[i] - 1];
    }
}

void generateLeftRightKey(char key[], char leftKey[], char rightKey[])
{
    int i;

    for (i = 0; i < 5; i++)
    {
        leftKey[i] = key[i];
    }

    for (i = 5; i < 10; i++)
    {
        rightKey[i - 5] = key[i];
    }
}

void leftShift(char key[], int shift)
{
    int i;
    char tempKey[5];

    strcpy(tempKey, key);

    for (i = 0; i < 5; i++)
    {
        key[i] = tempKey[(i + shift) % 5];
    }
}

void p8(char key[], char key1[])
{
    int i;
    int permutationTable[8] = {6, 3, 7, 4, 8, 5, 10, 9};
```

```
        for (i = 0; i < 8; i++)
        {
            key1[i] = key[permutationTable[i] - 1];
        }

        key1[i] = '\0';
    }

    void generateKey1(char leftKey[], char rightKey[], char key1[])
    {
        int i;
        char tempKey[10];

        leftShift(leftKey, 1);
        leftShift(rightKey, 1);

        for (i = 0; i < 5; i++)
        {
            tempKey[i] = leftKey[i];
        }

        for (i = 5; i < 10; i++)
        {
            tempKey[i] = rightKey[i - 5];
        }

        p8(tempKey, key1);
    }

    void generateKey2(char leftKey[], char rightKey[], char key2[])
    {
        int i;
        char tempKey[10];

        leftShift(leftKey, 2);
        leftShift(rightKey, 2);

        for (i = 0; i < 5; i++)
        {
            tempKey[i] = leftKey[i];
        }

        for (i = 5; i < 10; i++)
        {
            tempKey[i] = rightKey[i - 5];
        }

        p8(tempKey, key2);
    }

    void generateKeys(char key[], char key1[], char key2[])
    {
        char leftKey[5], rightKey[5], permutedKey[10];
        permuteKey(key, permutedKey);
        generateLeftRightKey(permutedKey, leftKey, rightKey);
        generateKey1(leftKey, rightKey, key1);
        generateKey2(leftKey, rightKey, key2);
    }
```

Encrypt.h

```
// HEADER FILE FOR ENCRYPTION

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

void charToBinary(char input, char *output)
{
    int i;
    if (input == 0) {
        for (i = 0; i < 8; i++)
        {
            output[i] = '0';
        }
    }
    else {
        for (i = 8; i > 0; i--)
        {
            if (input % 2 == 0)
            {
                output[i - 1] = '0';
            }
            else
            {
                output[i - 1] = '1';
            }
            input = input / 2;
        }
    }
    output[8] = '\0';
}

void binaryToChar(char *input, char *output)
{
    int i;
    *output = 0;
    for (i = 0; i < 8; i++)
    {
        *output = *output + (input[i] - '0') * pow(2, 7 - i);
    }
}

void ip8(char *input, char *output)
{
    int i;
```

```
        int ip[] = {2, 6, 3, 1, 4, 8, 5, 7};
        for (i = 0; i < 8; i++)
        {
            output[i] = input[ip[i] - 1];
        }

        output[8] = '\0';
    }

    void devide(char *input, char *left, char *right)
    {
        int i;
        for (i = 0; i < 4; i++)
        {
            left[i] = input[i];
            right[i] = input[i + 4];
        }

        left[4] = '\0';
        right[4] = '\0';
    }

    void ep(char *input, char *output)
    {
        int i;
        int ep[] = {4, 1, 2, 3, 2, 3, 4, 1};
        for (i = 0; i < 8; i++)
        {
            output[i] = input[ep[i] - 1];
        }

        output[8] = '\0';
    }

    void xor8bit(char *input1, char *input2, char *output)
    {
        int i;
        for (i = 0; i < 8; i++)
        {
            output[i] = (input1[i] - '0') ^ (input2[i] - '0') + '0';
        }

        output[8] = '\0';
    }

    void sBox(char *input, char *output)
    {
        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 i = (input[0] - '0') * 2 + (input[3] - '0');
        int j = (input[1] - '0') * 2 + (input[2] - '0');
        int k = (input[4] - '0') * 2 + (input[7] - '0');
        int l = (input[5] - '0') * 2 + (input[6] - '0');

        output[0] = s0[i][j] / 2 + '0';
        output[1] = s0[i][j] % 2 + '0';
        output[2] = s1[k][l] / 2 + '0';
        output[3] = s1[k][l] % 2 + '0';

        output[4] = '\0';
    }

    void p4(char *input, char *output)
    {
        int i;
        int p[] = {2, 4, 3, 1};
        for (i = 0; i < 4; i++)
        {
            output[i] = input[p[i] - 1];
        }

        output[4] = '\0';
    }

    void xor4bit(char *input1, char *input2, char *output)
    {
        int i;
        for (i = 0; i < 4; i++)
        {
            output[i] = (input1[i] - '0') ^ (input2[i] - '0') + '0';
        }

        output[4] = '\0';
    }

    void combine(char *input1, char *input2, char *output)
    {
        int i;
        for (i = 0; i < 4; i++)
        {
            output[i] = input1[i];
            output[i + 4] = input2[i];
        }

        output[8] = '\0';
    }

    void swap(char *input, char *output)
    {
        int i;
        char temp;
```

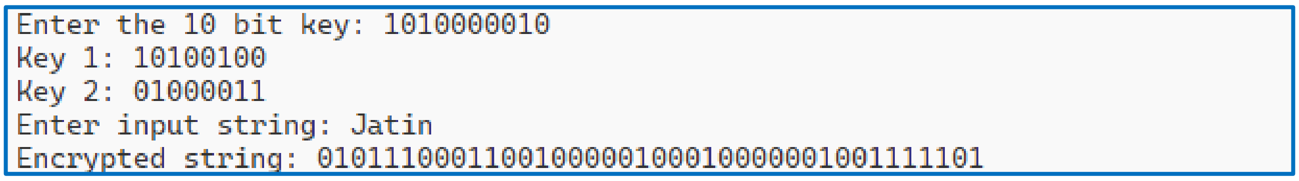
```
        for (i = 0; i < 4; i++)
        {
            temp = input[i];
            input[i] = output[i];
            output[i] = temp;
        }

        output[4] = '\0';
    }

    void ip8Inverse(char *input, char *output)
    {
        int i;
        int ip[] = {4, 1, 3, 5, 7, 2, 8, 6};
        for (i = 0; i < 8; i++)
        {
            output[i] = input[ip[i] - 1];
        }

        output[8] = '\0';
    }

    void encryptForK(char *bin, char *key, char *output)
    {
        // Step 2: IP8
        char ip8Output[9];
        ip8(bin, ip8Output);

        // Step 3: Devide 4n4
        char ip8Left[5], ip8Right[5];
        devide(ip8Output, ip8Left, ip8Right);

        // Step 4: EP
        char epOutput[9];
        ep(ip8Right, epOutput);

        // Step 5: XOR 8-bit
        char xorOutput[9];
        xor8bit(epOutput, key, xorOutput);

        // Step 6 and 7: S-Box
        char sBoxOutput[5];
        sBox(xorOutput, sBoxOutput);

        // Step 8: P4
        char p4Output[5];
        p4(sBoxOutput, p4Output);

        // Step 9:
        char xorOutput2[5];
        xor4bit(p4Output, ip8Right, xorOutput2);

        // Step 10: Combine S3
        char combineOutput[9];
```

```c
    combine(ip8Left, xorOutput2, combineOutput);

    // Step 11: Devide 4n4
    char combineLeft[5], combineRight[5];
    devide(combineOutput, combineLeft, combineRight);

    // Step 12: Swap
    swap(combineLeft, combineRight);

    // Step 13: Combine step 12 and generate output
    char combineOutput2[9];
    combine(combineLeft, combineRight, combineOutput2);

    // Copy the combined output to the final output
    strcpy(output, combineOutput2);
}

void encrypt(char c, char *k1, char *k2, char *output)
{
    // Step 1: Char to Binary
    char bin[9];
    charToBinary(c, bin);
    char output1[9], output2[9], output3[9];

    encryptForK(bin, k1, output1);
    encryptForK(output1, k2, output2);

    // Final inverse permutation
    ip8Inverse(output2, output3);

    output3[8] = '\0';

    // Copy the final output to the provided output buffer
    strcpy(output, output3);
}
```

SDES.c

```c
// SDES implementation in C
// REFERENCE: https://www.c-sharpcorner.com/article/s-des-or-simplified-data-
encryption-standard/

#include <stdio.h>
#include <string.h>
#include "keygen.h"
#include "encrypt.h"

char key[10];
char key1[9];
char key2[9];

char inputString[100];
char output[9];
char outputBits[900];


void main()
{
    int i;
    printf("Enter the 10 bit key: ");
    scanf("%s", &key);

    generateKeys(key, key1, key2);

    printf("Key 1: %s\n", key1);
    printf("Key 2: %s\n", key2);

    printf("Enter input string: ");
    scanf("%s", &inputString);

    int len = strlen(inputString);

    printf("Encrypted string: ");

    for (i = 0; i < len; i++)
    {
        encrypt(inputString[i], key1, key2, output);
        for (int j = 0; j < 8; j++)
        {
            outputBits[i * 8 + j] = output[j];
        }
    }

    printf("%s\n", outputBits);
}
```

OUTPUT



```
Enter the 10 bit key: 1010000010
Key 1: 10100100
Key 2: 01000011
Enter input string: Jatin
Encrypted string: 0101110001100100000100010000001001111101
```