main.c

Run

Output

Clear

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

#define TOTAL_ROUNDS 16

// Sample 64-bit key (parity bits included)
char key[65] =
    "1010101010111011000010010001100000100111001
    1011011001100110111011";

// Permuted Choice 1 (PC1): 64-bit to 56-bit
    (removes parity)
int PC1[56] = {
    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,

```

```
DES Subkeys (Each = 24 bits from C + 24 from D):
K 1: 100001111000000001100111011001111110000110011111
K 2: 000011110000000011001110110011111100001100111110
K 3: 001111000000001100111010001111110000110011111010
K 4: 111100000000110011101011111111000011001111101000
K 5: 110000000011001110101100111100001100111110100011
K 6: 000000001100111010110000110000110011111010001100
K 7: 000000110011101011000011000011001111101000110011
K 8: 000011001110101100001111001100111110100011001111
K 9: 000110011101011000011110011001111101000110011111
K10: 011001110101100001111000100111110100011001111110
K11: 100111010110000111100000011111010001100111111000
K12: 011101011000011110000000111101000110011111100001
K13: 110101100001111000000001110100011001111110000110
K14: 010110000111100000000110010001100111111000011001
K15: 011000011110000000011001000110011111100001100111
K16: 110000111100000000110011001100111111000011001111
```

main.c

Run

Output

Clear

```c
// Left shift schedule per round
int shift_schedule[16] = {
    1, 1, 2, 2, 2, 2, 2, 2,
    1, 2, 2, 2, 2, 2, 2, 1
};

// Left circular shift for 28-bit halves
void leftShift(char *half, int shifts) {
    char temp[28];
    for (int i = 0; i < 28; i++)
        temp[i] = half[(i + shifts) % 28];
    memcpy(half, temp, 28);
}

// Apply permutation
void permute(const char *input, char *output,
    const int *table, int size) {
    for (int i = 0; i < size; i++) {
```

```
DES Subkeys (Each = 24 bits from C + 24 from D):
K 1: 100001111000000001100111011001111110000110011111
K 2: 000011110000000011001110110011111100001100111110
K 3: 001111000000001100111010001111110000110011111010
K 4: 111100000000110011101011111111000011001111101000
K 5: 110000000011001110101100111100001100111110100011
K 6: 000000001100111010110000110000110011111010001100
K 7: 000000110011101011000011000011001111101000110011
K 8: 000011001110101100001111001100111110100011001111
K 9: 000110011101011000011110011001111101000110011111
K10: 011001110101100001111000100111110100011001111110
K11: 100111010110000111100000011111010001100111111000
K12: 011101011000011110000000111101000110011111100001
K13: 110101100001111000000001110100011001111110000110
K14: 010110000111100000000110010001100111111000011001
K15: 011000011110000000011001000110011111100001100111
K16: 110000111100000000110011001100111111000011001111
```

main.c

Run

```
        leftShift(D, shift_schedule[round]);

        // Build subkey: first 24 bits from C,
            last 24 bits from D
        strncpy(subkeys[round], C, 24);
        strncpy(subkeys[round] + 24, D, 24);
        subkeys[round][48] = '\0';
    }

    // Display subkeys
    printf("DES Subkeys (Each = 24 bits from C +
        24 from D):\n");
    for (int i = 0; i < TOTAL_ROUNDS; i++) {
        printf("K%2d: %s\n", i + 1, subkeys[i]);
    }

    return 0;
}
```

Output

Clear

```
K 2: 000011110000000011001110110011111100001100111110
K 3: 001111000000001100111010001111110000110011111010
K 4: 111100000000110011101011111111000011001111101000
K 5: 110000000011001110101100111100001100111110100011
K 6: 000000001100111010110000110000110011111010001100
K 7: 000000110011101011000011000011001111101000110011
K 8: 000011001110101100001111001100111111010001100111
K 9: 000110011101011000011110011001111110100011001111
K10: 011001110101100001111000100111110100011001111110
K11: 100111010110000111100000011111010001100111111000
K12: 011101011000011110000000111101000110011111100001
K13: 110101100001111000000001110100011001111110000110
K14: 010110000111100000000110010001100111111000011001
K15: 011000011110000000011001000110011111100001100111
K16: 110000111100000000110011001100111111000011001111

=== Code Execution Successful ===
```