SESSION DESPRICTIVE PROTOCOL

NAME: SINCHANA R

REG. NO: BU22EECE0100441

1.Write a program to count no. of bits which are set in given binary pattern2

```
#include <stdio.h>
// Function to count set bits in an integer
int countSetBits(int n) {
    int count = 0;
    while (n) {
        count += n & 1; // Increment count if the last bit is set
        n >>= 1; // Right shift n by 1 to check the next bit
    }
    return count;
}
int main() {
    int num;
    // Input the binary number
    printf("Enter an integer: ");
    scanf("%d", &num);
    // Call the function to count set bits
    int result = countSetBits(num);
    // Print the result
    printf("Number of set bits: %d\n", result);
    return 0;
}
```

2.Write a program to set 5th and 12th bits in a 16-bit unsigned integer.

```
def set_bits(n):
    # Set the 5th bit (index 4, 0-based)
    n |= (1 << 4)
    # Set the 12th bit (index 11, 0-based)
    n |= (1 << 11)
    return n
```

3.Write a program to clear 6th and 19th bits in a 32-bit unsigned integer.

```
#include <stdio.h>
unsigned int clearBits(unsigned int num) {
    unsigned int mask = ~(1 << 6 | 1 << 19);
    return num & mask;
}
int main() {
    unsigned int num;
    // Input the integer
    printf("Enter a 32-bit unsigned integer: ");
    scanf("%u", &num);
    // Clear the 6th and 19th bits
    unsigned int result = clearBits(num);
    // Print the result
    printf("Result after clearing 6th and 19th bits: %u\n", result);
    return 0;
}
```

4.Write a program to flip even positioned bits in a 16-bit unsigned integer

An IP Address will be in the form of "a.b,c.d" format, where a,b,c,d will be in the range of 0-255. Given a,b,c,d values (or string format) pack them into 32-bit unsigned integer.

Flip Even Positioned Bits in a 16-bit Unsigned Integer

```
#include <stdio.h>
int main() {
    unsigned short num;
    // Input the 16-bit unsigned integer
    printf("Enter a 16-bit unsigned integer: ");
    scanf("%hu", &num);
    // Flip even positioned bits
    for (int i = 0; i < 16; i += 2) {
        num ^= (1 << i);
    }
    // Print the result
    printf("Result after flipping even positioned bits: %hu\n", num);
    return 0;
}
```

Pack an IP Address into a 32-bit Unsigned Integer.

```
#include <stdio.h>
unsigned int packIPAddress(unsigned char a, unsigned char b, unsigned char c, unsigned char d)
{
    unsigned int ip = 0;
    ip |= (a << 24);
    ip |= (b << 16);
    ip |= (c << 8);
    ip |= d;
    return ip;
}

int main() {
    unsigned char a, b, c, d;
    // Input the IP address
    printf("Enter IP address in the format a.b.c.d: ");
    scanf("%hhu.%hhu.%hhu.%hhu", &a, &b, &c, &d);
    // Pack the IP address
    unsigned int packedIP = packIPAddress(a, b, c, d);
    // Print the packed IP address
    printf("Packed IP address: %u\n", packedIP);
    return 0;
}
```

5.Given an unsigned 32-bit integer holding packed IPv4 address, convert it into "a.b.c.d" format.

```
#include <stdio.h>
void unpackIPAddress(unsigned int ip, unsigned char *a, unsigned char *b, unsigned char *c,
unsigned char *d) {
    *a = (ip >> 24) & 255;
    *b = (ip >> 16) & 255;
    *c = (ip >> 8) & 255;
    *d = ip & 255;
}
int main() {
    unsigned int packedIP;
    unsigned char a, b, c, d;
    // Input the packed IP address
    printf("Enter packed IP address: ");
    scanf("%u", &packedIP);

    // Unpack the IP address
    unpackIPAddress(packedIP, &a, &b, &c, &d);
    // Print the unpacked IP address in "a.b.c.d" format
    printf("Unpacked IP address: %u.%u.%u.%u\n", a, b, c, d);
    return 0;
}
```

6.Convert MAC address into 48-bit binary pattern.

```
#include <stdio.h>
#include <stdint.h>
// Function to convert MAC address string to 48-bit binary pattern
uint64_t convertMACToBinary(const char *mac) {
    uint64_t binaryMAC = 0;
    unsigned int bytes[6];
    // Parse the MAC address string
    sscanf(mac, "%x:%x:%x:%x:%x:%x", &bytes[0], &bytes[1], &bytes[2], &bytes[3], &bytes[4], &bytes[5]);
    // Combine the bytes into a 48-bit integer
    for (int i = 0; i < 6; ++i) {
        binaryMAC = (binaryMAC << 8) | bytes[i];
    }
    return binaryMAC;
}
int main() {
    char mac[18];
    // Input the MAC address
    printf("Enter MAC address (format: XX:XX:XX:XX:XX:XX): ");
    scanf("%17s", mac);
    // Convert MAC address to 48-bit binary pattern
    uint64_t binaryMAC = convertMACToBinary(mac);
    // Print the binary pattern
    printf("48-bit binary pattern: %012llx\n", binaryMAC);
    return 0;
}
```

7.Convert 48 bit binary pattern as MAC address.

```
#include <stdio.h>
#include <stdint.h>
// Function to convert 48-bit binary pattern to MAC address string
void convertBinaryToMAC(uint64_t binaryMAC, char *mac) {
    sprintf(mac, "%02llx:%02llx: %02llx:%02llx:%02llx:%02llx",
            (binaryMAC >> 40) & 0xFF,
            (binaryMAC >> 32) & 0xFF,
            (binaryMAC >> 24) & 0xFF,
            (binaryMAC >> 16) & 0xFF,
            (binaryMAC >> 8) & 0xFF,
            binaryMAC & 0xFF);
}
int main() {
    uint64_t binaryMAC;
    char mac[18];
    // Input the 48-bit binary pattern
    printf("Enter 48-bit binary pattern (in hex): ");
    scanf("%llx", &binaryMAC);
    // Convert binary pattern to MAC address
    convertBinaryToMAC(binaryMAC, mac);
    // Print the MAC address
    printf("MAC address: %s\n", mac);
    return 0;
}
```

FLOWCHART FOR 7 PROGRAMS

1. Count Number of Set Bits in a Given Binary Pattern

Flowchart:

1. Start
2. Input: binary number `n`
3. Initialize: `count = 0`
4. While `n` is not zero:
    a. Increment `count` by `n & 1`
    b. Right shift `n` by 1 bit
5. Output: `count`
6. End

2. Set 5th and 12th Bits in a 16-bit Unsigned Integer

Flowchart:

7. Start
8. Input: 16-bit unsigned integer `num`
9. Set the 5th bit: `num |= (1 << 5)`

```
10.     Set the 12th bit: num |= (1 << 12)
11.     Output: num
12. End
```

3. Clear 6th and 19th Bits in a 32-bit Unsigned Integer

Flowchart:

```
13. Start
14.     Input: 32-bit unsigned integer num
15.     Clear the 6th bit: num &= ~ (1 << 6)
16.     Clear the 19th bit: num &= ~ (1 << 19)
17.     Output: num
18. End
```

4. Flip Even Positioned Bits in a 16-bit Unsigned Integer

Flowchart:

```
19. Start
20.     Input: 16-bit unsigned integer num
21. For i = 0 to 15 with step 2:
        a. Flip the i-th bit: num ^= (1 << i)
22.     Output: num
23. End
```

5. Pack IP Address into 32-bit Unsigned Integer

Flowchart:

```
24. Start
25.     Input: a, b, c, d
26.     Initialize: ip = 0
27.     Pack a: ip |= (a << 24)
28.     Pack b: ip |= (b << 16)
29.     Pack c: ip |= (c << 8)
30.     Pack d: ip |= d
31.     Output: ip
32. End
```

6. Unpack 32-bit Unsigned Integer into IP Address

Flowchart:

33. Start
34. Input: 32-bit unsigned integer `ip`
35. Extract `a`: `a = (ip >> 24) & 0xFF`
36. Extract `b`: `b = (ip >> 16) & 0xFF`
37. Extract `c`: `c = (ip >> 8) & 0xFF`
38. Extract `d`: `d = ip & 0xFF`
39. Output: `a.b.c.d`
40. End

7. Convert MAC Address into 48-bit Binary Pattern

Flowchart:

41. Start
42. Input: MAC address string `mac`
43. Parse `mac` into 6 bytes: `byte[0]` to `byte[5]`
44. Initialize: `binaryMAC = 0`
45. For each byte from `0` to `5`:
    a. Shift `binaryMAC` left by 8 bits
    b. OR `binaryMAC` with `byte[i]`
46. Output: `binaryMAC`
47. End

8. Convert 48-bit Binary Pattern into MAC Address

Flowchart:

48. Start
49. Input: 48-bit binary pattern `binaryMAC`
50. Extract bytes:
    a. `byte [0] = (binaryMAC >> 40) & 0xFF`
    b. `byte [1] = (binaryMAC >> 32) & 0xFF`
    c. `byte [2] = (binaryMAC >> 24) & 0xFF`
    d. `byte [3] = (binaryMAC >> 16) & 0xFF`
    e. `byte [4] = (binaryMAC >> 8) & 0xFF`
    f. `byte[5] = binaryMAC & 0xFF`

```
51.     Format and Output:
   byte[0]:byte[1]:byte[2]:byte[3]:byte[4]:byte[5]
52. End
```