**REPORT On Bitwise Operators Assignment Problems**

1. Write a program to count number of bits which are set in given binary pattern.

#include <stdio.h>

#define COUNT\_SET\_BITS(num) ({ int count = 0, n = (num); while (n) { count += n & 1; n >>= 1; } count; })

int main() {

    int number;

    printf("Enter an integer: ");

    scanf("%d", &number);

    int setBitsCount = COUNT\_SET\_BITS(number);

    printf("The number of set bits in the binary representation of %d is: %d\n", number, setBitsCount);

    return 0;

}

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

#include <stdio.h>

#define SET\_BIT(num, bit) ((num) | (1 << (bit)))

#define SET\_5TH\_AND\_12TH\_BITS(num) (SET\_BIT(SET\_BIT((num), 4), 11))

int main() {

    unsigned int number;

    printf("Enter a 16-bit unsigned integer: ");

    scanf("%u", &number);

    if (number > 0xFFFF) {

        printf("The number exceeds 16-bit range. Please enter a valid 16-bit unsigned integer.\n");

        return 1;

    }

    number = SET\_5TH\_AND\_12TH\_BITS(number);

    printf("The number after setting the 5th and 12th bits is: %u\n", number);

    return 0;

}

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

#include <stdio.h>

unsigned int clearBits(unsigned int num, int pos1, int pos2) {

    unsigned int mask1 = ~(1 << pos1);

    unsigned int mask2 = ~(1 << pos2);

    unsigned int combinedMask = mask1 & mask2;

    num &= combinedMask;

    return num;

}

void printBinary(unsigned int num) {

    for (int i = 31; i >= 0; i--) {

        printf("%u", (num >> i) & 1);

        if (i % 4 == 0) printf(" ");

    }

    printf("\n");

}

int main() {

    unsigned int num;

    int pos1, pos2;

    printf("Enter a 32-bit unsigned integer\n Eg: 25789458\n ");

    scanf("%u", &num);

    printf("Entered number in binary: ");

    printBinary(num);

    printf("Enter the position of the first bit to clear (0-31): ");

    scanf("%d", &pos1);

    printf("Enter the position of the second bit to clear (0-31): ");

    scanf("%d", &pos2);

    if (pos1 < 0 || pos1 > 31 || pos2 < 0 || pos2 > 31) {

        printf("Error: Bit positions must be between 0 and 31.\n");

        return 1;

    }

    unsigned int result = clearBits(num, pos1, pos2);

    printf("Resulting integer after clearing bits: %u\n", result);

    printf("Resulting number in binary: ");

    printBinary(result);

    return 0;

}

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

#include <stdio.h>

#define FLIP\_EVEN\_BITS(num) ((num) ^ 0x5555)

int main() {

    unsigned int number;

    printf("Enter a 16-bit unsigned integer: ");

    scanf("%u", &number);

    if (number > 0xFFFF) {

        printf("The number exceeds the 16-bit range. Please enter a valid 16-bit unsigned integer.\n");

        return 1;

    }

    number = FLIP\_EVEN\_BITS(number);

    printf("The number after flipping even-positioned bits is: %u\n", number);

    return 0;

}

1. 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 abcd values (or string format) pack them into 32-bit unsigned integer.

#include <stdio.h>

unsigned int packIP(unsigned char a, unsigned char b, unsigned char c, unsigned char d) {

    return (a << 24) | (b << 16) | (c << 8) | d;

}

int main() {

    unsigned char a, b, c, d;

    char ip\_address[16];

    printf("Enter the IP address (format: 172.22.208.98): ");

    scanf("%15s", ip\_address);

    sscanf(ip\_address, "%hhu.%hhu.%hhu.%hhu", &a, &b, &c, &d);

    unsigned int packedIP = packIP(a, b, c, d);

    printf("Packed IP address: 0x%X\n", packedIP);

    return 0;

}

1. Given an unsigned 32-bit integer holding packed into IPV4 address, convert it into “a.b.c.d” format.

#include <stdio.h>

#define BYTE1(ip) ((ip) >> 24) & 0xFF

#define BYTE2(ip) ((ip) >> 16) & 0xFF

#define BYTE3(ip) ((ip) >> 8) & 0xFF

#define BYTE4(ip) (ip) & 0xFF

int main() {

    unsigned int ip;

    printf("Enter the packed 32-bit unsigned integer (IP address): ");

    scanf("%u", &ip);

    printf("The IP address is: %u.%u.%u.%u\n", BYTE1(ip), BYTE2(ip), BYTE3(ip), BYTE4(ip));

    return 0;

}

1. Convert MAC Address into 48-bit binary pattern.

#include <stdio.h>

void macToBinary(const char\* mac) {

    unsigned int bytes[6];

    char binary[49];

    binary[48] = '\0';

    sscanf(mac, "%x:%x:%x:%x:%x:%x", &bytes[0], &bytes[1], &bytes[2], &bytes[3], &bytes[4], &bytes[5]);

    for (int i = 0; i < 6; ++i) {

        for (int bit = 0; bit < 8; ++bit) {

            binary[i \* 8 + (7 - bit)] = (bytes[i] & (1 << bit)) ? '1' : '0';

        }

    }

    printf("Binary representation: %s\n", binary);

}

int main() {

    char mac\_address[18];

    printf("Example of a MAC address: 2C-3B-70-58-1E-0B\n");

    printf("Enter a MAC address: ");

    scanf("%17s", mac\_address);

    macToBinary(mac\_address);

    return 0;

1. Convert 48-bit binary pattern as MAC Address.

#include <stdio.h>

#include <stdlib.h>

void binaryToMac(const char\* binary) {

    unsigned int bytes[6] = {0};

    for (int i = 0; i < 48; ++i) {

        bytes[i / 8] = (bytes[i / 8] << 1) | (binary[i] - '0');

    }

    printf("MAC Address: %02X:%02X:%02X:%02X:%02X:%02X\n",

           bytes[0], bytes[1], bytes[2], bytes[3], bytes[4], bytes[5]);

}

int main() {

    char binary\_pattern[49];

    printf("Example of a 48-bit binary pattern: 101010101011101111001100110111011110111111111111\n");

    printf("Enter a 48-bit binary pattern: ");

    scanf("%48s", binary\_pattern);

    binaryToMac(binary\_pattern);

    return 0;

}