VARSHINI.V

BU22EECE00100206

Assignment on Bitwise operators:-

1. **Write a program to count no. of bits which are set in given binary pattern 2 ?**

   Code:-

```
def count_set_bits(binary_pattern):
    count = 0
    for bit in binary_pattern:
        if bit == '1':
            count += 1
    return count

binary_pattern = input("Enter a binary pattern: ")
print("Number of set bits:", count_set_bits(binary_pattern))
```

   Output:-

```
Enter a binary pattern: 101010101
Number of set bits: 5
```

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

   Code:-

```
def set_bits(integer, *positions):
    for pos in positions:
        integer |= (1 << pos)
    return integer
```

```
# Initialize a 16-bit unsigned integer
unsigned_integer = 0b0000000000000000

# Set the 5th and 12th bits
unsigned_integer = set_bits(unsigned_integer, 5, 12)

print("Resulting integer with set bits:", bin(unsigned_integer))
```

Output:-

Resulting integer with set bits: 0b100010000000

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

Code:-

```
def clear_bits(integer, *positions):
    for pos in positions:
        integer &= ~(1 << pos)
    return integer

# Initialize a 32-bit unsigned integer
unsigned_integer = 0b11111111111111111111111111111111

# Clear the 6th and 19th bits
unsigned_integer = clear_bits(unsigned_integer, 6, 19)
```

Output:-

Resulting integer with cleared bits:
0b11111011111111111111111111101111

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.**

Code:-

```
def flip_even_bits(integer):
    for i in range(0, 16, 2):
        integer ^= (1 << i)
```

```
    return integer

def pack_ip_address(a, b, c, d):
    ip_address = (a << 24) | (b << 16) | (c << 8) | d
    return ip_address

# Input IP address
a, b, c, d = map(int, input("Enter IP Address in format 'a.b.c.d': ").split('.'))

# Pack IP address into a 32-bit unsigned integer
ip_integer = pack_ip_address(a, b, c, d)

# Flip even positioned bits in the packed IP address
flipped_integer = flip_even_bits(ip_integer)

print("Packed IP Address (before flipping even bits):", bin(ip_integer))
print("Packed IP Address (after flipping even bits):", bin(flipped_integer))
```

Output:-

```
Enter IP Address in format 'a.b.c.d': 192.168.1.1
Packed IP Address (before flipping even bits):
0b11000000101010000000000100000001
Packed IP Address (after flipping even bits):
0b11000000001010000000001100000001
```

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

Code:-

```
def unpack_ip_address(ip_integer):
    a = (ip_integer >> 24) & 255
    b = (ip_integer >> 16) & 255
    c = (ip_integer >> 8) & 255
    d = ip_integer & 255
    return a, b, c, d

# Input packed IP address
```

```
ip_integer = int(input("Enter packed IP Address (32-bit unsigned
integer): "))

# Convert packed IP address into "a.b.c.d" format
a, b, c, d = unpack_ip_address(ip_integer)

print("IP Address in 'a.b.c.d' format:", "{}.{}.{}.{}".format(a, b, c, d))
```

Output:-

```
IP Address in 'a.b.c.d' format: 192.168.1.1
```

6. Convert MAC address into 48 bit binary pattern ?

Code:-

```
def mac_to_binary(mac_address):
    binary_pattern = ''
    for part in mac_address.split(':'):
        binary_part = bin(int(part, 16))[2:].zfill(8)  # Convert each part to
binary and ensure it's 8 bits long
        binary_pattern += binary_part
    return binary_pattern

# Input MAC address
mac_address = input("Enter MAC address (in format xx:xx:xx:xx:xx:xx): ")

# Convert MAC address into 48-bit binary pattern
binary_pattern = mac_to_binary(mac_address)

print("Binary pattern of MAC address:", binary_pattern)
```

Output:-

```
Binary pattern of MAC address:
000000001101101010110011010010111100110111110
```

7. Convert 48 bit binary pattern as MAC address ?

Code:-

```
def binary_to_mac(binary_pattern):
    # Check if the binary pattern length is exactly 48
    if len(binary_pattern) != 48:
        return "Invalid binary pattern length. It should be 48 bits."

    # Split the binary pattern into 6 groups of 8 bits each
    groups = [binary_pattern[i:i+8] for i in range(0, 48, 8)]

    # Convert each group from binary to hexadecimal
    hex_groups = [hex(int(group, 2))[2:].zfill(2) for group in groups]

    # Concatenate the hexadecimal groups with colons to form the MAC
address
    mac_address = ":".join(hex_groups)

    return mac_address

# Example usage
binary_pattern =
"110000001010100000001111001100101100100011011001"
mac_address = binary_to_mac(binary_pattern)
print("MAC Address:", mac_address)
```

Output:-

MAC Address: C0:A8:0F:32:C8:D9