# Bitwise Operators Assignment

Q1. WAP to read a 8 bit unsigned integer, interchange the adjacent bits i.e D0  with D1, D2 with D3….. D6 with D7.  Display the final number.

Input: 0xAA

Output: 0x55

Ans:

#include <stdio.h>

int main() {

unsigned char num, result;

printf("Enter an 8-bit unsigned integer (in hexadecimal): ");

scanf("%hhx", &num);

result = ((num & 0xAA) >> 1) | ((num & 0x55) << 1);

printf("Input: 0x%X\n", num);

printf("Output: 0x%X\n", result);

return 0;

}

Q2. WAP to count the number of 1’s in a given byte and display

Ans:

#include <stdio.h>

int main() {

unsigned char num;

int count = 0;

printf("Enter an 8-bit unsigned integer (in hexadecimal): ");

scanf("%hhx", &num);

while (num) {

count += num & 1;

num >>= 1;

}

printf("Number of 1's: %d\n", count);

return 0;

}

Q3. Generate odd and even parity bits for a given number. (consider a 32 bit number)

[Hint: You may reuse the solution created in Q2 and extend it further]

Ans:

#include <stdio.h>

int count\_ones(unsigned int num) {

int count = 0;

while (num) {

count += num & 1;

num >>= 1;

}

return count;

}

int main() {

unsigned int num;

int ones\_count, even\_parity, odd\_parity;

printf("Enter a 32-bit unsigned integer (in hexadecimal): ");

scanf("%x", &num); // Reads a 32-bit number in hexadecimal format

ones\_count = count\_ones(num);

// Even Parity: Parity bit to make the count of 1's even

if (ones\_count % 2 == 0) {

even\_parity = 0;

} else {

even\_parity = 1;

}

// Odd Parity: Parity bit to make the count of 1's odd

if (ones\_count % 2 == 0) {

odd\_parity = 1;

} else {

odd\_parity = 0;

}

printf("Input: 0x%X\n", num);

printf("Even Parity Bit: %d\n", even\_parity);

printf("Odd Parity Bit: %d\n", odd\_parity);

return 0;

}

Q4. WAP to reverse the bytes in a 32 but unsigned integer using shift operator.

Input: 0x12345678

Output: 0x78563412

Ans:

#include <stdio.h>

int main() {

unsigned int num, reversed;

printf("Enter a 32-bit unsigned integer (in hexadecimal): ");

scanf("%x", &num); // Reads a 32-bit number in hexadecimal format

// Reverse the bytes using bitwise operations

reversed = ((num & 0xFF) << 24) |

((num & 0xFF00) << 8) |

((num & 0xFF0000) >> 8) |

((num & 0xFF000000) >> 24);

printf("Input: 0x%X\n", num);

printf("Reversed: 0x%X\n", reversed);

return 0;

}