1) 0x000000ff tem todos os bits do byte menos significativo iguais a 1 e os outros bytes todos em 0, então ao fazer esse valor & x, o resultado vai ter os bits do byte menos significativo iguais a 1 se os de x forem 1 e iguais a 0 se os de x forem 0; todo o restante será 0.

0xff000000 tem todos os bits do byte mais significativo iguais a 1 e o restante igual a 0. O operador | retorna 1 quando um dos bits é 1 e 0 caso contrário. Portanto, 0xff000000 | x retorna 1 para todos os bits do byte mais significativo, e para os outros bytes retorna 1 se o bit de x for 1 e 0 se o bit de x for 0.

2) dentro do for, a função verifica se o bit menos significativo de x é 1 e se for incrementa uma variável contadora, depois desloca x para a direita. Quando todos os bits de x estão em zero, sai do for e verifica se o bit menos significativo da variável contadora é 1; se sim, esse valor é ímpar.

3)

a) unsigned char alto tem os bits 4, 5, 6, e 7 iguais aos bits 0, 1, 2 e 3 de x e o restante igual a 0; unsigned char baixo tem os bits 0, 1, 2 e 3 iguais aos bits 4, 5, 6 e 7 de x e o restante igual a 0. Usando o operador |, a função retorna um valor que tem os bits 0, 1, 2 e 3 iguais aos de baixo (porque alto tem esses bits iguais a 0 e 0 | y = y) e os bits 4, 5, 6 e 7 iguais aos de alto (porque baixo tem esses bits iguais a 0 e 0 | y = y).

b) guardei o valor de x deslocado n bits para a esquerda em um unsigned char alto; essa variável tem os n bits menos significativos iguais a 0. Unsigned char baixo tem os seus n bits menos significativos iguais aos n bits mais significativos de x, e o restante igual a 0. Assim, usando o operador | os n bits menos significativos de alto ficam iguais aos de baixo, que são os bits deslocados de x.